

Incorporación de Mecanismos de Usabilidad en un Entorno de Producción de Software Dirigido por Modelos

Jose Ignacio Panach Navarrete



Tesis Doctoral



UNIVERSIDAD
POLITECNICA
DE VALENCIA

Directores:
Óscar Pastor López
Natalia Juristo Juzgado

Mayo 2010

Incorporación de Mecanismos de Usabilidad en un Entorno de Producción de Software Dirigido por Modelos

Jose Ignacio Panach Navarrete

Centro de Investigación en Métodos de Producción de Software

Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia



UNIVERSIDAD
POLITECNICA
DE VALENCIA

Director: Óscar Pastor López

CoDirectora: Natalia Juristo Juzgado

Mayo 2010

Dedicatoria

A mis padres, Rosa y Matías:

Por apoyarme en todos mis años de estudio

A mi hermano, Jorge:

Por hacer el día a día lo más feliz posible

A mi novia, Lorena:

Por su paciencia, su comprensión y su amor

Agradecimientos

Esta tesis es fruto del trabajo de investigación realizado durante los últimos años dentro del centro de investigación *Métodos de Producción de Software (PROS)*. Es para mi un gran honor expresar mi gratitud a sus miembros y colaboradores, los cuales me han ayudado sin dudarle en todo momento.

Mención especial requiere mi director, compañero y amigo, Óscar Pastor. Gracias a él entré a formar parte del centro PROS en el año 2005 (antiguo grupo de investigación OO-Method) y sin su apoyo esta tesis no se hubiera podido llevar a cabo.

También quiero agradecer de manera especial la ayuda de mi codirectora Natalia Juristo, de la Universidad Politécnica de Madrid (UPM). Las reuniones celebradas han sido de gran impulso para este trabajo. Además, su esfuerzo por supervisar todos los temas tratados en la tesis ha sido el mismo que si no existiera distancia entre Valencia y Madrid.

Para elaborar el capítulo de evaluación de la propuesta ha sido necesaria la colaboración de mucha gente. En primer lugar, quiero agradecer las ideas de Nelly Condori dentro del grupo PROS y de Sira Vegas de la UPM. Ambas me han ayudado a diseñar el experimento. En segundo lugar, también ha sido vital la participación de los 66 voluntarios anónimos que realizaron el experimento, prestándome una pequeña parte de su valioso tiempo. Por último, quiero agradecer la ayuda y el interés de Alejandro Catalá en el análisis de los datos.

Quiero también agradecer a las personas y organismos que han financiado mi actividad investigadora en estos últimos años. En primer lugar, dar las gracias a Arturo González del Río y Ana Gadea por la oportunidad que me dieron de entrar a trabajar en el departamento DSIC como becario del Máster de Ingeniería del Software. También quiero dar las gracias a la empresa CARE Technologies, y en especial a José Miguel Barberá, por financiarme una beca de especialización durante unos meses. Por último, agradecer a la Generalitat Valenciana la concesión de una beca FPI durante 4 años.

También me gustaría mencionar a los compañeros del laboratorio 1L04 del DSIC que han hecho las jornadas de trabajo amenas y han demostrado que somos amigos dentro y fuera de la universidad. Han sido una fuente de inspiración para mi trabajo de investigación y me han ayudado sin vacilar en numerosas ocasiones: Francisco Valverde, Sergio España, Nathalie Aquino, Jose Luis de la Vara, Nelly Condori, Marcela Ruiz y Urko Rueda. También quiero mencionar a otros miembros del PROS que de manera desinteresada me han apoyado a lo largo de mi trabajo: Bea, Giovanni, Pau, Estefanía, Ana C., Paqui, Carlos, Ismael, Nacho, Miriam, Pablo, María, Mario, Joan, Pele, Juan, Manoli, Tanja, Arthur, Pedro, Vicky, Jorge, Jose Vicente, Ana L., Sergio S., Ainhoa, Matthijs, M^a José, Ana M. y Verónica. Por último, también quiero recordar a aquellos compañeros que por unas u otras razones ya no trabajan en el PROS: Inés Pederiva, Marta Ruiz, Luis Iñesta, David Melo y David Anés.

Resumen

Actualmente, uno de los principales retos de la Ingeniería del Software (IS) es el desarrollo de sistemas de calidad. La calidad es una propiedad del software que cuenta con diversas características, entre ellas la usabilidad, que es en la que se centra esta tesis.

La IS se ha centrado históricamente en problemas de funcionalidad y de persistencia, relegando a un segundo plano aspectos de la interacción con el usuario, y más concretamente, de la usabilidad. Este vacío ha sido cubierto por la comunidad Interacción Persona-Ordenador (IPO), que ha propuesto recomendaciones para mejorar la usabilidad. Algunas de estas recomendaciones deben ser consideradas desde las primeras fases de construcción de los sistemas a fin de evitar realizar cambios en la arquitectura una vez ésta haya sido diseñada. Estas recomendaciones se conocen como *Functional Usability Features (FUF)*.

La incorporación de los FUFs desde las primeras fases del proceso de desarrollo añade cierta complejidad a la construcción de sistemas, ya que el analista ha de tener en cuenta más factores a la hora del desarrollo. Esta tesis presenta una solución basada en transformaciones entre modelos. El objetivo de la tesis es el de presentar un método (llamado MIMAT) para incorporar los FUFs dentro de un método de desarrollo *Model-Driven Development (MDD)*. Para ello, se profundiza en los cambios que el diseñador del método MDD debe aplicar para enriquecer dicho método con los FUFs. Una vez incorporados los FUFs, el analista que modele sistemas con el método MDD puede utilizar las características de usabilidad en el desarrollo de sistemas. La principal ventaja de esta aproximación es que el analista puede incorporar las características de usabilidad simplemente a partir de modelos conceptuales, dejando la implementación a las transformaciones de modelo a código.

La factibilidad de MIMAT se ha demostrado aplicándolo a un método de desarrollo MDD específico: OO-Method. Además, el esfuerzo que debería realizar el diseñador del método MDD para incluir los FUFs ha sido justificado por medio de un experimento empírico. Este experimento ha demostrado la mejora en la satisfacción del usuario tras incluir los FUFs.

Resum

Actualment, un del principal reptes de l' Enginyeria del Maquinari (EM) és la construcció de sistemes de qualitat. La qualitat és una propietat del programari que està formada per diverses característiques, entre les quals trobem la usabilitat, en la qual està centrada aquesta tesi.

L' EM ha estat centrada històricament en problemes de funcionalitat i persistència, oblidant la interacció amb el usuari i més concretament, la usabilitat. Aquest buit ha sigut cobert per la comunitat Interacció Persona-Ordinador (IPO), que ha proposat recomanacions per millorar la usabilitat dels sistemes. Algunes d'estes recomanacions han de ser considerades des de les primeres etapes de la construcció del sistemes per tal d'evitar realitzar canvis en la arquitectura una volta aquesta haja sigut dissenyada. Aquestes recomanacions s'anomenen *Functional Usability Features (FUF)*.

La incorporació dels FUFs des de les primeres etapes del procés de desenvolupament afegeix certa complexitat a la construcció de sistemes, perquè l'analista ha de tindre en compte més factors a l'hora del desenvolupament. Aquesta tesi presenta una solució basada en la transformació de models. L'objectiu de la tesi és el de presentar un mètode (anomenat MIMAT) per incorporar els FUFs dins d'un mètode de desenvolupament *Model-Driven Development (MDD)*. Per tant, la tesi aprofundeix en els canvis que el dissenyador del mètode MDD cal que aplique per tal d'enriquir-lo amb els FUFs. Una vegada incorporats els FUFs, l'analista que modele amb el mètode MDD modificat pot utilitzar les característiques d'usabilitat en el desenvolupament de sistemes. El principal avantatge d'aquesta aproximació és que l'analista pot incorporar les característiques d'usabilitat simplement a partir dels models conceptuals, mentres que deixa la implementació a les transformacions de model a codi.

La factibilitat de MIMAT s'ha demostrat aplicant-lo a un mètode de desenvolupament MDD específic: OO-Method. A més a més, l'esforç del dissenyador del mètode MDD per tal d'incloure els FUFs ha sigut justificat per mitjà d'un experiment empíric. Aquest experiment ha demostrat que la satisfacció de l'usuari millora després d'incloure FUFs en el procés de desenvolupament del programari.

Abstract

Currently, one of the most important challenges in the Software Engineering (SE) community is the development of quality systems. Quality is a software property that is composed of different characteristics. From all these characteristics, this PhD thesis focuses on usability.

SE focused historically on problems related to functionality and persistence, pushing into the background interaction issues and more specifically, usability. This gap has been fixed by the Human-Computer Interaction (HCI), which has proposed a set of recommendations to improve usability in systems. Some of these recommendations must be considered from the very early software development steps in order to prevent changes in the system architecture once this architecture has been designed. This type of recommendations is called Functional Usability Features (FUF).

The incorporation of FUFs from the early steps of the software development process increases the complexity of systems construction because the analyst has to deal with more issues. This PhD thesis presents a solution based on model transformations. The goal of this thesis is to define a method (called MIMAT) to include FUFs in a Model-Driven Development (MDD) software development process. For this aim, the document studies in depth the changes that the MDD designer must apply to the MDD method in order to enrich the software development process with FUFs. Once FUFs has been included, the analyst that works with the MDD method can use those new usability features to develop future systems. The main advantage of this proposal is that the analyst can include usability features by means of conceptual models, relegating the code generation to model-code transformations.

MIMAT has been applied to a specific MDD method, called the OO-Method, in order to demonstrate its feasibility. Moreover, the effort of the MDD designer in order to include FUFs has been justified by means of an empirical experiment. This experiment concludes that the user's satisfaction improves after including FUFs in the software development process.

Índice

Índice	1
1 Introducción	9
1.1 Área de investigación	9
1.2 Descripción del problema.....	11
1.3 Aproximación a la solución.....	16
1.4 Estructura de la tesis	22
PARTE I: ANTECEDENTES Y PROBLEMA A RESOLVER	25
2 Estado de la Cuestión	27
2.1 Incorporación de la usabilidad en la arquitectura del sistema	29
2.1.1 Definición de la declaración de la usabilidad del sistema	29
2.1.1.1 Descripción.....	29
2.1.1.2 Ventajas y limitaciones de la propuesta.....	31
2.1.2 Relación de usabilidad con la arquitectura a través de escenarios.....	32
2.1.2.1 Descripción.....	32
2.1.2.2 Ventajas y limitaciones de la propuesta.....	35
2.1.3 Framework de usabilidad definido por Folmer.....	35
2.1.3.1 Descripción.....	35
2.1.3.2 Ventajas y limitaciones de la propuesta.....	38
2.1.4 Conclusiones	38
2.2 Incorporación de la usabilidad a nivel de requisitos.....	39
2.2.1 Estilos para la captura de requisitos propuestos por Lauesen. 40	
2.2.1.1 Descripción.....	40
2.2.1.2 Ventajas y limitaciones de la propuesta.....	43
2.2.2 Propuesta para incorporar requisitos de calidad al proceso de desarrollo	44
2.2.2.1 Descripción.....	44
2.2.2.2 Ventajas y limitaciones de la propuesta.....	46
2.2.3 Captura de los requisitos de usabilidad mediante i*.....	47
2.2.3.1 Descripción.....	47
2.2.3.2 Ventajas y limitaciones de la propuesta.....	50
2.2.4 Propuesta de modelado del usuario y de la usabilidad de Adikari	50
2.2.4.1 Descripción.....	50
2.2.4.2 Ventajas y limitaciones de la propuesta.....	53
2.2.5 Captura de requisitos de mecanismos de usabilidad	54
2.2.5.1 Descripción.....	54

2.2.5.2	Ventajas y limitaciones de la propuesta.....	56
2.2.6	Conclusiones	57
2.3	Patrones de usabilidad.....	58
2.3.1	Patrones de usabilidad de Perzel para ambientes Web.....	58
2.3.1.1	Definición.....	58
2.3.1.2	Ventajas y limitaciones de la propuesta.....	61
2.3.2	Patrones de interacción de Welie	61
2.3.2.1	Definición.....	61
2.3.2.2	Ventajas y limitaciones de la propuesta.....	64
2.3.3	Representación formal de patrones de usabilidad de Borchers.....	64
2.3.3.1	Definición.....	64
2.3.3.2	Ventajas y limitaciones de la propuesta.....	66
2.3.4	Colección de patrones de usabilidad de Brighton	67
2.3.4.1	Descripción.....	67
2.3.4.2	Ventajas y limitaciones de la propuesta.....	69
2.3.5	Patrones de diseño de la interacción de Tidwell	69
2.3.5.1	Descripción.....	69
2.3.5.2	Ventajas y limitaciones de la propuesta.....	71
2.3.6	Representación de los patrones de usabilidad mediante un repositorio de ontologías de Henninger.....	72
2.3.6.1	Definición.....	72
2.3.6.2	Ventajas y limitaciones de la propuesta.....	74
2.3.7	Patrones de interacción para el trabajo cooperativo	75
2.3.7.1	Descripción.....	75
2.3.7.2	Ventajas y limitaciones de la propuesta.....	77
2.3.8	Conclusiones	78
2.4	Usabilidad en entornos MDD	79
2.4.1	Modelado de la usabilidad con un Diagrama de Transición entre Estados	80
2.4.1.1	Definición.....	80
2.4.1.2	Ventajas y limitaciones de la propuesta.....	81
2.4.2	Evaluación temprana de la usabilidad en MDA.....	82
2.4.2.1	Descripción.....	82
2.4.2.2	Ventajas y limitaciones de la propuesta.....	85
2.4.3	Conclusiones	87
2.5	Visión general del estado de la cuestión.....	88
3	Planteamiento del Problema	97
3.1	Problema a resolver	97
3.2	Contribuciones con respecto al estado de la cuestión.....	98
3.3	Aproximación a la solución.....	100
3.3.1	Funcionalidades de usabilidad.....	100
3.3.1.1	El proyecto STATUS	100
3.3.1.2	Mecanismos de usabilidad.....	103
3.3.1.3	Ventajas de los mecanismos de usabilidad con respecto a otras propuestas.....	109
3.4	Tecnologías de transformación de modelos	111

3.4.1	Ventajas e inconvenientes de las tecnologías de transformación de modelos.....	119
3.4.2	Una tecnología de transformación de modelos: OO-Method .	124
3.4.2.1	Computation Independent Model (CIM)	125
3.4.2.2	Platform Independent Model (PIM)	126
3.4.2.3	Platform Specific Model (PSM)	130
3.4.2.4	Code Model (CM)	131
3.4.2.5	Ámbito de las aplicaciones generadas por OO-Method...	137
PARTE II: RESOLUCIÓN.....		139
4 Método de Incorporación de Mecanismos de Usabilidad a una TTM: MIMAT.....		141
4.1	Definición de formas de uso.....	144
4.2	Identificación de propiedades.....	145
4.2.1	Propiedades configurables	146
4.2.2	Propiedades no configurables	146
4.3	Definición de primitivas conceptuales	147
4.4	Cambios necesarios en el compilador de modelos.....	149
4.5	Incorporación de mecanismos de usabilidad a OO-Method	152
4.6	Caso de ilustración	154
5 Incorporación de Feedback.....		159
5.1	System Status Feedback	159
5.1.1	Formas de uso	159
5.1.2	Propiedades	163
5.1.3	Modificación de los modelos conceptuales de OO-Method ...	176
5.1.3.1	WU_SSF1: Informar del éxito o fracaso en la ejecución del servicio	183
5.1.3.2	WU_SSF2: Mostrar el estado de la información	185
5.1.3.3	WU_SSF3: Mostrar el estado de las acciones.....	186
5.1.4	Modificación del compilador de modelos.....	189
5.1.4.1	WU_SSF1: Informar del éxito o fracaso en la ejecución del servicio	189
5.1.4.2	WU_SSF2: Mostrar el estado de la información	191
5.1.4.3	WU_SSF3: Mostrar el estado de las acciones.....	193
5.1.4.4	Resumen de los cambios que provoca System Status Feedback	195
5.2	Interaction Feedback	198
5.2.1	Formas de uso	198
5.2.2	Propiedades.....	199
5.2.3	Modificación de los modelos conceptuales de OO-Method ...	202
5.2.4	Modificación del compilador de modelos.....	203
5.2.4.1	WU_IF1: Informar que la interacción está siendo atendida	203

5.2.4.2	Resumen de los cambios que provoca Interaction Feedback	205
5.3	Progress Feedback	206
5.3.1	Formas de uso	206
5.3.2	Propiedades	208
5.3.3	Modificación de los modelos conceptuales de OO-Method ...	212
5.3.3.1	WU_PF1: Mostrar progreso de la ejecución	215
5.3.4	Modificación del compilador de modelos	217
5.3.4.1	WU_PF1: Mostrar progreso de la ejecución	217
5.3.4.2	Resumen de los cambios que provoca Progress Feedback..	221
5.4	Warning	222
5.4.1	Formas de uso	222
5.4.2	Propiedades	224
5.4.3	Modificación de los modelos conceptuales de OO-Method ...	227
5.4.3.1	WU_W1: Mensaje de aviso	228
5.4.4	Modificación del compilador de modelos	230
5.4.4.1	WU_W1: Mensaje de aviso	230
5.4.4.2	Resumen de los cambios que provoca Warning	233
6	Incorporación de Wizard.....	235
6.1	Step by Step	235
6.1.1	Formas de uso	235
6.1.2	Propiedades	236
6.1.3	Modificación de los modelos conceptuales de OO-Method ...	244
6.1.3.1	WU_SBS1: Definir un asistente	247
6.1.4	Modificación del compilador de modelos	249
6.1.4.1	WU_SBS1: Definir un asistente	249
6.1.4.2	Resumen de los cambios que provoca Step by Step	251
7	Incorporación de User Input Error Prevention	253
7.1	Structured Text Entry	253
7.1.1	Formas de uso	253
7.1.2	Propiedades	256
7.1.3	Modificación de los modelos conceptuales de OO-Method ...	260
7.1.3.1	WU_STE1: Especificar el tipo de visualización del campo de entrada	261
7.1.4	Modificación del compilador de modelos	263
7.1.4.1	WU_STE1: Tipo de campo de entrada	263
7.1.4.2	Resumen de los cambios que provoca Structured Text Entry	264
8	Incorporación de User Profile	267
8.1	Preferences	267

8.1.1	Formas de uso	267
8.1.2	Propiedades	269
8.1.3	Modificación de los modelos conceptuales de OO-Method ...	274
8.1.3.1	WU_P1: Preferencias en el aspecto visual	278
8.1.4	Modificación del compilador de modelos	281
8.1.4.1	WU_P1: Preferencias en el aspecto visual	281
8.1.4.2	Resumen de los cambios que provoca Preferences	284
8.2	Personal Object Space	287
8.2.1	Formas de uso	287
8.2.2	Propiedades	288
8.2.3	Modificación de los modelos conceptuales de OO-Method ...	291
8.2.3.1	WU_POS1: Distribución de elementos	293
8.2.4	Modificación del compilador de modelos	296
8.2.4.1	WU_POS1: Distribución de elementos	296
8.2.4.2	Resumen de los cambios que provoca Personal Object Space	299
8.3	Favourites	301
8.3.1	Formas de uso	301
8.3.2	Propiedades	302
8.3.3	Modificación de los modelos conceptuales de OO-Method ...	306
8.3.3.1	WU_F1: Definición de favoritos	308
8.3.4	Modificación del compilador de modelos	309
8.3.4.1	WU_F1: Definición de favoritos	309
8.3.4.2	Resumen de los cambios que provoca Favourites	311
9	Incorporación de Undo y Cancel	313
9.1	Global Undo	313
9.1.1	Formas de uso	313
9.1.2	Propiedades	315
9.1.3	Modificación de los modelos conceptuales de OO-Method ...	320
9.1.3.1	WU_GU1: Deshacer cambios	324
9.1.3.2	WU_GU2: Rehacer cambios	326
9.1.4	Modificación del compilador de modelos	327
9.1.4.1	WU_GU1: Deshacer cambios	327
9.1.4.2	WU_GU2: Rehacer cambios	331
9.1.4.3	Resumen de los cambios que provoca Global Undo	334
9.2	Abort Operation	337
9.2.1	Formas de uso	337
9.2.2	Propiedades	338
9.2.3	Modificación de los modelos conceptuales de OO-Method ...	343
9.2.3.1	WU_AO1: Cancelar durante la ejecución	344
9.2.3.2	WU_AO2: Salir de un escenario	346
9.2.4	Modificación del compilador de modelos	347
9.2.4.1	WU_AO1: Cancelar durante la ejecución	347
9.2.4.2	WU_AO2: Salir de un escenario	349
9.2.4.3	Resumen de los cambios que provoca Abort Operation ..	350

10 Incorporación de Help.....	353
10.1 Multilevel Help	353
10.1.1 Formas de uso	353
10.1.2 Propiedades.....	355
10.1.3 Modificación de los modelos conceptuales de OO-Method ...	360
10.1.3.1 WU_MH1: Ayuda dinámica	363
10.1.3.2 WU_MH2: Ayuda estática	365
10.1.4 Modificación del compilador de modelos.....	366
10.1.4.1 WU_MH1: Ayuda dinámica	366
10.1.4.2 WU_MH2: Ayuda estática	368
10.1.4.3 Resumen de los cambios que provoca Multilevel Help....	368
 PARTE III: EVALUACIÓN Y CONCLUSIONES.....	 371
11 Evaluación de Viabilidad	373
11.1 Construcción del Modelo de Objetos	374
11.1.1 Modelado de WU_SSF1	375
11.1.2 Modelado de WU_GU1 y WU_GU2	376
11.1.3 Modelado de WU_AO1	377
11.1.4 Modelado de WU_STE3	378
11.1.5 Modelado de WU_W1	379
11.2 Construcción del Modelo de Interacción Abstracto.....	381
11.2.1 Modelado de WU_SBS1	382
11.2.2 Modelado de WU_PF1.....	385
11.2.3 Modelado de WU_SSF2 y WU_SSF3	386
11.2.4 Modelado de WU_STE2	387
11.2.5 Modelado de WU_MH1.....	388
11.2.6 Modelado de GU1 y GU2.....	389
11.2.7 Modelado de WU_MH2.....	392
11.2.8 Modelado de WU_F1	393
11.3 Construcción del Modelo de Interacción Concreto	394
11.3.1 Modelado de WU_SBS1	395
11.3.2 Modelado de WU_STE1	396
11.3.3 Modelado de WU_P1	397
11.3.4 Modelado de WU_POS1.....	400
11.3.5 Modelado de WU_AO1	401
11.3.6 Modelado de WU_AO2	403
11.3.7 Modelado de WU_MH1.....	404
11.3.8 Modelado de WU_PF1.....	405
11.3.9 Modelado de WU_SSF1	407
11.3.10 Modelado de WU_SSF2	410
11.3.11 Modelado de WU_W1	410
11.3.12 Modelado de WU_F1	411
11.4 Conclusiones de la viabilidad	412

12 Evaluación Experimental	415
12.1 Formas de uso utilizadas en el experimento.....	416
12.2 Diseño del experimento.....	422
12.2.1 Objetivos.....	422
12.2.2 Preguntas y métricas.....	423
12.2.3 Planificación detallada.....	424
12.2.4 Diseño del experimento.....	426
12.2.5 Proceso del experimento.....	427
12.2.6 Amenazas a la validez.....	429
12.3 Relación entre formas de uso y atributos de usabilidad.....	430
12.3.1 Atributos de System Status Feedback.....	431
12.3.2 Atributos de Warning.....	432
12.3.3 Atributos de Structured Text Entry.....	433
12.3.4 Atributos de Multilevel Help.....	434
12.4 Instrumentos utilizados en el experimento.....	435
12.4.1 Cuestionario demográfico.....	435
12.4.2 Tareas y cuestionario.....	436
12.4.2.1 Tarea 1.....	437
12.4.2.2 Tarea 2.....	439
12.4.2.3 Tarea 3.....	441
12.4.2.4 Tarea 4.....	442
12.4.2.5 Cuestionario final.....	443
12.5 Aplicaciones Web utilizadas en el experimento.....	444
12.5.1 Tarea 1: Crear coche.....	445
12.5.2 Tarea 2: Crear cuenta bancaria.....	447
12.5.3 Tarea 3: Hacer una reserva de alquiler.....	449
12.5.4 Tarea 4: Poner coche en venta.....	450
12.6 Análisis de datos.....	452
12.6.1 Hipótesis sobre la satisfacción del usuario ($H1_0$):.....	453
12.6.1.1 Estudio de las medias.....	453
12.6.1.2 Estudio del ANOVA.....	464
12.6.1.3 Gráficos box and whiskers.....	475
12.6.1.4 Conclusiones del análisis de la satisfacción.....	479
12.6.2 Hipótesis sobre la eficiencia del usuario ($H2_0$):.....	480
12.6.2.1 Estudio de las medias.....	480
12.6.2.2 Estudio del ANOVA.....	484
12.6.2.3 Gráficos box and whiskers.....	488
12.6.2.4 Conclusiones del análisis de la eficiencia.....	490
12.6.3 Conclusiones de la evaluación experimental.....	490
13 Conclusiones y Trabajos Futuros	493
Referencias	499
Anexo I: Mecanismos de Usabilidad	507

Anexo II: Datos Recogidos en el Experimento	533
Índice de Figuras.....	545
Índice de Tablas	555

Capítulo 1

Introducción

Este capítulo presenta de forma breve y resumida, cuál es el ámbito en el que se desarrolla esta tesis, qué problema trata de resolver, un esbozo de la solución que propone y cómo ha sido validada.

1.1 Área de investigación

Esta tesis se encuadra dentro de la línea de investigación que intenta reducir el hueco entre la usabilidad y la Ingeniería del Software (IS). Según el estándar ISO/IEC 9126-1 [ISO/IEC 9126-1, 2001], la **usabilidad** se define como:

La capacidad del producto software de ser entendido, aprendido y agradable para el usuario bajo ciertas condiciones específicas de uso.

En el ámbito del desarrollo de software dirigido por modelos, la IS se ha centrado fundamentalmente en el modelado de aspectos funcionales y de persistencia de información de los sistemas, relegando a un segundo plano el modelado de las interfaces de usuario y más concretamente de la usabilidad. Pero actualmente la usabilidad está pasando a encabezar la lista de factores estratégicos para conseguir el éxito en el desarrollo de sistemas software [Thibodeau, 2002][IBM, 2005]. Según la ISO/IEC 9126-1, la usabilidad es una de las características que determinan la calidad de los sistemas (junto con la funcionalidad, seguridad, eficiencia, mantenibilidad, y portabilidad). Por tanto, aquellos sistemas que no sean usables no serán de calidad, aunque satisfagan todos los requisitos de funcionalidad impuestos por el usuario. Según Folmer [Folmer, 2004], los sistemas con mucha

funcionalidad pero difíciles de usar no podrán tener éxito en el mercado. Es preferible desarrollar sistemas con poca funcionalidad pero que ésta sea usable.

El vacío dejado por la comunidad de la IS en aspectos de usabilidad ha sido cubierto en su mayoría por los trabajos desarrollados dentro de la comunidad de Interacción Persona Ordenador (IPO). Esta comunidad está formada por expertos en el pensamiento humano, en el aprendizaje y en psicología pero que carecen habitualmente conocimientos informáticos a nivel de analista de sistemas. Sus propuestas se centran en determinar cómo deben ser las interfaces de los sistemas software para conseguir sistemas usables, pero dejando de lado el resto de aspectos que componen los sistemas (el resto de características de calidad). De ahí que exista un abismo entre las propuestas de las comunidades IS e IPO y que ambas propuestas se complementen, es decir, lo que no cubre una comunidad lo cubre la otra. **Esta tesis se enmarca en la línea de investigación que trata de acercar ambas comunidades unificando sus propuestas en soluciones que engloben tanto aspectos de usabilidad como funcionales.** En nuestra investigación, tomamos las mejoras de usabilidad propuestas por la comunidad IPO y las adaptamos e enriquecemos a un proceso de desarrollo software definido por la IS.

De entre todos los procesos de desarrollo de la IS, esta tesis está enmarcada en el proceso ***Model-Driven Development (MDD)***, que traducimos por Desarrollo de sistemas Dirigidos por Modelos [Mellor, 2002]. Este método de desarrollo propone el uso de modelos conceptuales para representar los sistemas de forma abstracta. Gracias a una serie de transformaciones, previamente definidas, el analista puede generar el sistema representado a partir de los modelos conceptuales. La *Object Management Group (OMG)* ha propuesto un estándar MDD que se llama ***Model Driven Architecture (MDA)*** [MDA, 2008]. En este estándar se definen una serie de modelos donde se representa el sistema software a distintos niveles de abstracción que van desde lo más abstracto posible (la representación del sistema sin tener en cuenta su computación) hasta el código que implementa el sistema.

La principal ventaja de MDD es que el analista es más eficiente en el desarrollo de sistemas, ya que sólo debe modelar el sistema y es el proceso

de transformación el encargado de su implementación. El grado de automatización del proceso de transformación de modelos va desde el cien por cien de automatización, hasta que el analista tenga que aplicar las transformaciones de manera manual. Cuanto más se automatice el proceso, menos trabajo debe realizar el analista. Las **Tecnologías de Transformación de Modelos (TTM)** surgen dentro del ámbito MDD proponiendo el desarrollo de sistemas completamente funcionales a partir de transformaciones automáticas entre modelos de distintos niveles de abstracción. Por lo tanto abogan por el desarrollo de sistemas a partir de modelos conceptuales. La tesis se engloba dentro de esta tecnología ya que persigue el mismo fin: la generación de código que represente un sistema usable.

En resumen, lo que se pretende con esta tesis es **proponer un método mediante el cual incorporar las características definidas por la IPO para conseguir sistemas usables a cualquier entorno de desarrollo MDD.**

1.2 Descripción del problema

Esta tesis propone una solución a parte del problema causado por el hueco existente entre las comunidades IPO e IS. Concretamente aborda el problema de cómo incorporar la usabilidad promovida por la comunidad IPO dentro de los procesos de producción de software más avanzados propuestos en el ámbito de la comunidad IS.

Históricamente, la IS ha tendido a separar el diseño de la interfaz del resto de etapas del proceso de desarrollo software, argumentando que las interfaces se podían diseñar en las últimas etapas del desarrollo independientemente de la funcionalidad del sistema. De ahí que esta comunidad se haya centrado en los aspectos funcionales de los sistemas software. Al relegar la IS el diseño de la interfaz, también ha quedado relegado a las últimas etapas del proceso de desarrollo software la incorporación de características que doten al sistema de usabilidad. Por otro lado, han proliferado los trabajos en la comunidad IPO que están relacionados con el diseño de interfaces y guías de usabilidad que deben seguir las interfaces. Esta comunidad ha definido una serie de

recomendaciones que debería incluir todo sistema software para poder considerarlo como usable. Desde la perspectiva de IS, estas recomendaciones se pueden dividir en tres grandes grupos, dependiendo de su naturaleza [Juristo, 2007, a]:

- Recomendaciones con impacto en la interfaz de usuario: Estas recomendaciones se refieren a aspectos de presentación que requieren una leve modificación del diseño de la interfaz. En este grupo se incluyen recomendaciones sobre botones, fondos, menús, etc.
- Recomendaciones con impacto en el proceso de desarrollo: Estas recomendaciones no afectan a los sistemas, sino al proceso de desarrollo. Definen cómo debe ser el proceso de desarrollo para conseguir que los sistemas software producidos sean usables. Ejemplos de este tipo de recomendaciones son: involucrar al usuario en el proceso de desarrollo; reducir la carga cognitiva; conocer la forma de pensar de los usuarios, etc.
- Recomendaciones con impacto en el diseño: Estas recomendaciones implican añadir cierta funcionalidad en el sistema para mejorar la interacción entre éste y el usuario. Ejemplos de este tipo de recomendaciones son las acciones de cancelar, deshacer o adaptar la funcionalidad del sistema a las preferencias de los usuarios. Estas recomendaciones han sido denominadas **Functional Usability Features (FUF)**¹. Los FUFs se dividen en varios subtipos más especializados llamados **mecanismos de usabilidad**. Por ejemplo, el FUF deshacer se divide en los mecanismos de usabilidad: deshacer global, deshacer de un objeto y abortar operación.

La incorporación de las recomendaciones del primer tipo es sencilla incluso en las últimas etapas del desarrollo. En cuanto a las recomendaciones del segundo tipo, afectan a la globalidad del proyecto, no a unas fases

¹ En Castellano, Características de Usabilidad Funcionales

concretas. Ambos tipos de recomendaciones no afectan a la arquitectura del sistema, sino a las interfaces y a las tareas a realizar durante el desarrollo respectivamente. Sin embargo, hoy en día se ha demostrado ([Bass, 2003] [Folmer 2004] [Comstock, 1996] [Juristo, 2007, a]) que la incorporación de las recomendaciones del tercer tipo (FUF) afectan a la arquitectura de los sistemas. Por tanto, el tener en cuenta los FUFs en las últimas etapas del proceso de desarrollo, puede implicar la modificación de la arquitectura del sistema una vez ésta ya albergue toda la lógica de negocio del sistema. Estos cambios en la arquitectura son costosos y requieren un gran esfuerzo del analista, el cual puede que tenga que modificar gran parte del diseño de alto nivel para introducir en la arquitectura la usabilidad junto con la lógica de negocio.

Para superar esta dificultad y conseguir sistemas usables con el menor esfuerzo posible, algunos autores han propuesto tratar la usabilidad desde las primeras etapas del desarrollo software [Bass, 2003] [Folmer 2004] [Juristo, 2007, a]. De esta forma, se incluye en la arquitectura del sistema la usabilidad como si de un requisito más de la lógica de negocio se tratara. Esta estrategia ya se utiliza para el resto de características de calidad (funcionalidad, seguridad, eficiencia, mantenibilidad, y portabilidad), cuya incorporación al sistema sería impensable en las últimas etapas del proceso de desarrollo.

La solución de incorporar los FUFs en las primeras etapas del desarrollo hace que el analista minimice los cambios en la arquitectura del sistema para incorporar características de la usabilidad, ya que la arquitectura se construye desde un inicio teniendo en cuenta la usabilidad. Sin embargo, la incorporación de los FUFs en todas las etapas del desarrollo tiene una serie de inconvenientes:

- **Aumento de la complejidad en el desarrollo de sistemas:** La incorporación de la usabilidad a lo largo de todo el proceso de desarrollo del sistema incrementa la complejidad de las tareas que debe realizar el analista. Para sistemas muy grandes, donde los usuarios requieran gran cantidad de características de usabilidad, el tiempo que debe dedicar el analista a esta incorporación es muy elevado. Se han llevado a cabo estudios para averiguar el coste de incorporar algunos mecanismos de usabilidad en los que se dividen

los FUFs [Juristo, 2007, a]. Estos estudios revelan que el esfuerzo del analista para incorporar mecanismos con impacto en el diseño como *Cancel* o *Feedback* son muy grandes. Aunque estos mecanismos no añaden muchas clases nuevas, la complejidad de sus métodos y el número de interacciones que son necesarias con el resto de clases del sistema hacen que la complejidad de incorporación sea muy elevada. La Tabla 1.1 muestra los resultados del estudio de esfuerzo para incorporar ambos mecanismos. Para cada mecanismo se muestra el coste asociado a: la incorporación de su funcionalidad; al número de clases nuevas; a la complejidad de los métodos de las nuevas clases; y al número de interacciones necesarias con otros componentes del sistema para que los métodos funcionen. El coste tiene tres categorías: alta; media y baja.

	FEEDBACK	CANCEL
FUNCIONALIDAD	Alta (87%)	Alta (95%)
CLASES	Baja (3.3%)	Baja (10%)
COMPLEJIDAD DE MÉTODOS	Media (cálculo de tiempos, control de errores, etc.)	Alta (volver a pasos anteriores, guardar estados, salir de operaciones)
INTERACCIONES	Alta (76%)	Media/Alta (66%)

Tabla 1.1 Resultados del estudio de la complejidad de incorporar *Feedback* y *Cancel* a la arquitectura de un sistema

Según los resultados, la incorporación del mecanismo *Cancel* es ligeramente más costosa que la incorporación de *Feedback*.

- **Errores en la captura de requisitos arrastran el problema hasta la implementación:** Son comunes los casos en que el usuario no sabe expresar los requisitos del futuro sistema o que el analista interpreta mal estos requisitos [Kozima, 2005] [Leite, 1996]. En

estos casos, hasta que el sistema no se haya implementado no se detectarán los errores en la captura de requisitos. Para resolver los problemas de usabilidad una vez el sistema esté implementado hay que modificar la arquitectura del sistema, por lo que los cambios serán muy costosos. Por tanto, en cierto modo nos encontramos con el mismo problema que teníamos al incorporar tarde la usabilidad en el proceso de desarrollo.

- **Errores a lo largo del proceso de desarrollo:** En algunos casos el analista puede cometer errores en alguna de las fases de desarrollo [Potts, 1993] [Daniels, 2007] [Cortex, 2009]. Los requisitos pueden estar bien especificados pero un error del analista en la fase de análisis o diseño puede hacer que se tenga que modificar la arquitectura del sistema si el error se detecta una vez el sistema esté ya implementado.
- **Requisitos variables:** En algunos sistemas los requisitos pueden cambiar a lo largo del tiempo [Goguen, 1993][Chung, 1999][Lawrence, 2001]. Por ejemplo, una acción del sistema que no se considere crítica a la hora de tomar los requisitos puede pasar a serlo antes de que la implementación esté finalizada. Este cambio de requisitos implicaría cambios importantes en la arquitectura del sistema y el coste para incorporar estos cambios sería muy elevado.
- **Dependencia del lenguaje de implementación:** La forma en la que la usabilidad se introduce en el sistema es particular para cada uno de los sistemas desarrollados y depende de la tecnología de implementación sobre la que esté escrito el código. Dependiendo del lenguaje de implementación, la recomendación se incorporará a la arquitectura del sistema de una forma distinta. Por tanto, aunque se definan guías sobre cómo incorporar la usabilidad en la arquitectura, el analista tendrá que realizar el esfuerzo de adaptar estas guías a los casos particulares.
- **La implementación se realiza manualmente:** El analista debe ser capaz, a partir de los modelos de análisis y diseño, de implementar los mecanismos de usabilidad junto con el resto de la lógica de negocio del sistema. Para ello es requisito imprescindible que el

analista tenga conocimientos de implementación y no sólo de análisis.

- **La relación coste/beneficio no es siempre favorable:** Aunque hay estudios sobre el coste de incorporación de cada uno de los mecanismos de usabilidad al sistema, en algunos casos el coste puede ser superior al estimado. Hay una serie de factores que afectan en el coste de incorporación de los mecanismos de usabilidad y no se pueden prever fácilmente, como la destreza que tenga el analista a la hora de implementar los mecanismos de usabilidad; las facilidades que proporciona el lenguaje de programación sobre el que se implementa el sistema; el número de mecanismos de usabilidad que haya que incorporar; las relaciones entre los mecanismos de usabilidad a incorporar. Puede que el coste de incorporar los mecanismos de usabilidad al sistema sea tan elevado que no compense con el beneficio obtenido por dicha incorporación.

Esta tesis propone una solución para que el esfuerzo del analista no se vea incrementado al introducir características de usabilidad, aliviando muchos de los problemas mencionados. A continuación se esboza la solución que proponemos, que se desarrolla a lo largo de la tesis.

1.3 Aproximación a la solución

La solución que se propone en esta tesis para minimizar el esfuerzo debido a la introducción de características de usabilidad en todas las etapas del proceso de desarrollo va encaminada a automatizar la incorporación de la usabilidad. La automatización del proceso de incorporación de propiedades de usabilidad casa con el paradigma MDD, donde a partir de una serie de transformaciones, se puede construir un sistema software de forma automática a partir de modelos conceptuales.

Esta tesis propone un método para incluir propiedades de usabilidad a un método de desarrollo software dirigido por modelos o Model-Driven Development (MDD) [Mellor, 2003] y más concretamente a las Tecnologías de Transformación de Modelos (TTM). Se ha bautizado la solución

propuesta con el nombre de MIMAT (*Método de Incorporación de Mecanismos de usAbilidad a una TTM*). Los métodos MDD abogan por construir sistemas software a partir de modelos conceptuales mediante transformaciones, pero no especifican cómo llevar a cabo dichas transformaciones. Las TTMs aportan una solución a este problema proponiendo una serie de transformaciones entre modelos. Se pretende que el grado de automaticidad de las transformaciones sea lo más elevado posible, pero existen TTMs con niveles de automaticidad muy diversos. Esta tesis propone un método para incorporar la usabilidad dentro de las transformaciones utilizadas por las TTMs.

Siguiendo la tendencia de incorporar la usabilidad en las primeras etapas de desarrollo, esta tesis está particularmente centrada en los mecanismos de usabilidad (agrupados en FUFs) definidos por Juristo [Juristo, 2007, b]. La elección de estos mecanismos de usabilidad como base para la tesis ha estado motivada en que dicho trabajo es uno de los pocos que hemos encontrado en los que se propone la incorporación de la usabilidad a nivel de requisitos. Así, una vez definidos los requisitos, en la arquitectura se pueden incorporar los mecanismos de usabilidad al mismo tiempo que se incorpora la lógica de negocio. Incorporando la usabilidad a nivel de requisitos se minimizan los cambios que el analista debe realizar en la arquitectura para conseguir sistemas usables. Además, otra de las ventajas de estos mecanismos de usabilidad es que su descripción es lo suficientemente detallada como para extraer primitivas conceptuales que representen dichos mecanismos de manera abstracta.

La idea de incorporar la usabilidad desde la fase de captura de requisitos hasta la fase de implementación es compatible con la idea de producir software usando Tecnologías de Transformación de Modelos (TTMs). Las TTMs especifican una serie de transformaciones entre modelos que permiten construir sistemas desde las fases tempranas de desarrollo hasta la implementación del sistema. Muchas de estas transformaciones se pueden automatizar para facilitar la labor al analista y reducir el tiempo de construcción de los sistemas. Por lo tanto, introduciendo las propiedades de usabilidad en las primeras fases de desarrollo en el marco de una TTM, se garantiza que la usabilidad se tenga en cuenta a lo largo de todo el proceso de desarrollo de software. Además, el hecho de que se pueda automatizar el proceso facilita la incorporación de los mecanismos de usabilidad en el

sistema. El analista sólo tiene que introducirlos en las primeras fases del proceso de desarrollo mediante las primitivas conceptuales y las transformaciones hacen posible su tratamiento a lo largo de las siguientes fases.

Así, las soluciones que aporta el método propuesto con respecto a los inconvenientes anteriormente descritos son:

- La complejidad en las tareas que debe realizar el analista para incorporar los mecanismos de usabilidad en el sistema sólo aumenta en las primeras fases del desarrollo. Una vez modelados en las primeras fases, los mecanismos de usabilidad se pueden derivar de estas fases a las siguientes gracias a las transformaciones. Cuanto mayor sea el grado de automatismo de estas transformaciones, menor será el esfuerzo del analista.
- Los errores en la captura de requisitos de usabilidad o su modificación posterior siguen afectando a todo el proceso de desarrollo en una TTM, pero en este caso se pueden corregir sin mucho esfuerzo. El analista sólo debe modificar el sistema en la fase de la TTM en la que se especifican los mecanismos de usabilidad, ya que en el resto de fases los cambios en la especificación de los mecanismos de usabilidad se pueden propagar mediante transformaciones.
- El modelado de la usabilidad a nivel conceptual es totalmente independiente de plataforma. Por lo tanto, las características de usabilidad modeladas son utilizables tanto para aplicaciones de escritorio como Web y para cualquier lenguaje de programación. Es decir, un mismo modelado puede dar lugar a aplicaciones para distintas plataformas. Las reglas de transformación son las encargadas de adaptar las características de usabilidad a los requisitos de la plataforma destino.
- Los errores provocados por el analista se minimizan porque las transformaciones automáticas entre las distintas fases de desarrollo de la TTM hacen posible que el analista tenga que realizar menos trabajo manual para incorporar los mecanismos de usabilidad.

Es importante distinguir las dos figuras existentes en el desarrollo basado en TTM. Por un lado, está la figura del diseñador de la TTM. El diseñador se encarga de definir los modelos conceptuales que forman la TTM y las estrategias de generación de código en base a dichos modelos. Por otro lado, está la figura del analista que trabaja con la TTM para construir sistemas utilizando dicha tecnología. El analista es el encargado de modelar los sistemas utilizando el modelo conceptual de la TTM y generar el código a transformaciones de modelo a código. **El método MIMAT está pensado para que lo aplique el diseñador de la TTM. Una vez incorporados los mecanismos de usabilidad en la TTM, el analista podría utilizar las nuevas primitivas para conseguir generar aplicaciones más usables de las que se podían generar anteriormente.**

La base del método MIMAT consiste en distinguir entre las funcionalidades de los mecanismos, qué propiedades pueden representarse abstractamente con primitivas conceptuales y qué otras propiedades pueden estar embebidas dentro de la propia estrategia de transformación entre modelos. Las propiedades de los mecanismos que necesitan primitivas conceptuales son aquellas que requieren que el analista tome alguna decisión para configurar su funcionalidad, mientras que las propiedades que no necesitan primitivas conceptuales son aquellas que no tienen ninguna alternativa en su configuración, y por tanto, pueden incorporarse al sistema software sin que el analista deba aportar ninguna información. Esta división de los mecanismos de usabilidad es válida para cualquier TTM; en cambio, la definición de las primitivas conceptuales y los cambios en la estrategia de generación de código son específicos para una TTM, ya que cada TTM tiene un modelo conceptual y una estrategia de generación distintas.

Como ejemplo de TTM, esta tesis ha utilizado OO-Method [Pastor, 2007]. OO-Method es un método de desarrollo software que soporta el estándar MDA [MDA, 2008], ya que diferencia los distintos artefactos software del sistema en los diferentes niveles de abstracción definidos en la propuesta MDA. Por un lado representa el espacio del problema en base a modelos conceptuales, y por otro representa el espacio de la solución en base al código correspondiente en un lenguaje de programación específico. La elección de OO-Method para esta tesis con respecto al resto de TTM ha estado motivada por sus dos principales ventajas:

1. Genera aplicaciones totalmente funcionales a partir de modelos conceptuales sin que el analista tenga que escribir una sola línea de código.
2. Sus modelos conceptuales son lo suficientemente abstractos como para representar todos los aspectos de un sistema.
3. OO-Method tiene una implementación industrial en la herramienta OlivaNOVA [CARE, 2008], por lo que los resultados de esta tesis pueden tener adicionalmente una aplicación directa en el mundo de la industria si en un futuro se incorporan a OlivaNOVA. Esta herramienta es capaz de representar de forma gráfica los modelos conceptuales de OO-Method y generar el código que implementa lo expresado en dichos modelos. Este proceso de transformación se lleva a cabo mediante su compilador de modelos.

El proceso de incorporación de los mecanismos de usabilidad a una TTM implica la definición de nuevas primitivas conceptuales con las que representar la usabilidad. Por tanto, una de las contribuciones principales de esta tesis es la definición de las primitivas conceptuales que hacen falta para representar cada una de las propiedades de los mecanismos de usabilidad en OO-Method. Para ello se propone una extensión del modelo conceptual de OO-Method.

Las nuevas primitivas conceptuales, además de introducir los cambios pertinentes en el modelo conceptual de OO-Method y en la herramienta OlivaNOVA, implican lógicamente introducir cambios en la estrategia de generación de código, es decir, en el compilador de modelos. El compilador de modelos de OO-Method es el encargado de generar el código, en un lenguaje de programación específico que implementa el sistema representado en el modelo conceptual. El compilador de modelos debe ser capaz de reconocer las nuevas primitivas conceptuales derivadas de los mecanismos de usabilidad y generar el código que represente su funcionalidad tal y como la haya definido el analista.

Además de los cambios en el compilador de modelos derivados de las nuevas primitivas conceptuales, hay propiedades de los mecanismos de usabilidad que, aunque no tienen representación abstracta (primitivas

conceptuales), también implican cambios. Son aquellas propiedades de los mecanismos de usabilidad que no requieren de las decisiones del analista porque su valor viene impuesto por guías de usabilidad y por tanto se pueden embeber directamente en la estrategia de generación de código. El compilador de modelos debe ser capaz de generar también el código que implementa este tipo de propiedades. Los cambios en el compilador de modelos se hacen una única vez y son válidos para todos los sistemas desarrollados. Por tanto, el esfuerzo de realizar estos cambios se amortiza en cada uno de los sistemas generados.

Una vez definidos los cambios que se deben aplicar a OO-Method, el último objetivo de esta tesis es el de demostrar la factibilidad del método MIMAT y la mejora de usabilidad en las aplicaciones generadas tras la aplicación de dicho método. Por un lado, para demostrar la factibilidad, se han diseñado unos prototipos de ventanas desde los cuales se podrían modelar las nuevas primitivas conceptuales que representan las propiedades de los mecanismos de forma abstracta. Estos prototipos dan una idea de los pasos que tendría que seguir el analista de OO-Method para trabajar con los mecanismos de usabilidad. Aunque los prototipos están especialmente pensados para el modelo conceptual de OO-Method, también se pueden utilizar como ejemplo para guiar la aplicación de los mecanismos de usabilidad a otras TTMs.

Por otro lado, se realiza una validación experimental para comprobar si la usabilidad de las aplicaciones generadas con OO-Method mejora o no tras la aplicación del método MIMAT. Para ello se utilizan dos grupos de usuarios. Un grupo interactuó con una aplicación generada por OlivaNOVA sin incorporar la funcionalidad de los mecanismos de usabilidad, mientras que el otro grupo interactuó realizando las mismas tareas en la misma aplicación que el primer grupo pero esta vez incorporando los mecanismos de usabilidad. Este experimento pretende determinar la mejora en el grado de satisfacción y eficiencia que los usuarios experimentan en la aplicación que ha incorporado los mecanismos de usabilidad con respecto a la que no.

1.4 Estructura de la tesis

La investigación presentada en este documento se ha organizado en tres grandes bloques. En un primer bloque se estudia el estado del arte y el problema que se pretende resolver con la tesis. Este bloque contiene los capítulos 2 y 3. En un segundo bloque se presenta una propuesta para solucionar el problema planteado. Este bloque abarca del capítulo 4 al 10. En un tercer bloque se presenta la evaluación de la propuesta y las conclusiones. Este último bloque incluye los capítulos del 11 al 13. A continuación se va a detallar el contenido de cada uno de los capítulos:

- **Capítulo 2:** Discute otras propuestas que tratan la usabilidad desde las primeras etapas del proceso de desarrollo software. Concretamente, se comparan los trabajos que proponen incorporar la usabilidad a los sistemas de distintas maneras: desde la fase de diseño de la arquitectura del sistema; desde la fase de captura de requisitos; mediante el uso de patrones de diseño; dentro de un método de desarrollo MDD.
- **Capítulo 3:** Sienta las bases de la tesis. Para ello presenta los mecanismos de usabilidad sobre los que se fundamenta esta tesis. Además, define lo que se entiende por un método de desarrollo dirigido por modelos (MDD), haciendo hincapié en sus ventajas e inconvenientes con respecto a los métodos de desarrollo tradicionales. Dentro del ámbito de los métodos MDD, también define lo que se entiende por una Tecnología de Transformación de Modelos (TTM).
- **Capítulo 4:** Propone el método MIMAT para incorporar los mecanismos de usabilidad a cualquiera de las TTMs existentes. Además, se describe el caso de ilustración que se utiliza a modo de ejemplo a lo largo de los siguientes capítulos de la tesis.
- **Capítulo 5 a 10:** Explican respectivamente cómo se aplica el método MIMAT para incorporar a OO-Method los mecanismos de usabilidad de: *Feedback*; *Wizard*; *Structured Text Entry*; *User Profile*; *Undo y Cancel*; *Help*. Se detallan todos los cambios que se deben realizar en OO-Method, tanto a nivel conceptual como de

compilador de modelos. Para ejemplificar estos cambios se utiliza el caso de ilustración.

- **Capítulo 11:** Muestra cómo quedaría OO-Method tras la incorporación de los mecanismos de usabilidad presentados en los capítulos anteriores. Para ello se utilizan prototipos de ventanas desde las cuales se podría modelar cada una de las propiedades de los mecanismos de usabilidad. Este capítulo se utiliza para demostrar que el método MIMAT se puede aplicar a una TTM real como es OO-Method.
- **Capítulo 12:** Expone la validación experimental realizada para comprobar si la usabilidad de las aplicaciones generadas por OO-Method mejora o no tras la aplicación del método MIMAT. Para ello, se presenta un diseño experimental y un análisis estadístico exhaustivo para extraer conclusiones de los resultados obtenidos en base a dicho diseño experimental.
- **Capítulo 13:** Discute las conclusiones y los trabajos futuros.

**PARTE I:
ANTECEDENTES Y
PROBLEMA A RESOLVER**

Capítulo 2

Estado de la Cuestión

Este capítulo revisa los trabajos existentes que tratan la usabilidad desde las primeras etapas del proceso de desarrollo software. En concreto se centra en:

- Los trabajos que incorporan la usabilidad durante la construcción de la arquitectura del sistema.
- Los trabajos que incorporan la usabilidad desde la fase de captura de requisitos.
- Los trabajos que han definido patrones de usabilidad para reutilizar una solución a los principales problemas de usabilidad en cualquier sistema.
- Los trabajos que tratan la usabilidad dentro de un método de desarrollo MDD.

A continuación se detallan los principales trabajos de cada una de estas áreas. Para cada uno de los trabajos estudiados se resaltan sus ventajas y sus limitaciones. Los criterios seguidos para realizar el estudio son los siguientes:

- **Tecnología y ambientes:** Valora las tecnologías o ambientes sobre los que se puede aplicar la propuesta. Por ejemplo, aplicaciones de escritorio, ambientes Web, aplicaciones en sistemas embebidos, etc.
- **Incluye pasos o guías:** Valora si la propuesta define una serie de pasos o guías que faciliten la labor del analista a la hora de tratar

con dicha propuesta. Por ejemplo, si la propuesta incluye una lista de buenas prácticas o un manual detallado para guiar al analista.

- **Método de desarrollo para el que fue pensada:** Indica el método de desarrollo sobre el que se podría aplicar la propuesta. Por ejemplo, si fue pensada para un proceso de desarrollo tradicional o para un método MDD.
- **Notación precisa para representar la usabilidad:** Indica si la propuesta incluye una notación clara para representar la usabilidad de forma abstracta. Por ejemplo, si usa una notación basada en patrones, o una notación basada en DTEs.
- **Especifica cómo implementar la propuesta:** Valora si la propuesta incluye una definición sobre cómo implementarla en un entorno real. Por ejemplo, los cambios que hay que hacer tanto en la arquitectura del sistema como en el código del lenguaje de programación.
- **Cantidad de atributos de usabilidad tratados:** Contabiliza los atributos de usabilidad que se tratan en la propuesta. Por ejemplo, en el caso de una propuesta basada en patrones se contarían los atributos de usabilidad que el autor haya relacionado con cada patrón.
- **Grado de comprensibilidad por el usuario:** Valora lo fácil o difícil que es entender la propuesta para el usuario. Este criterio de valoración es muy importante para aquellas propuestas que requieran la colaboración del usuario. Por ejemplo, si la propuesta usa prototipos, éstos son muy entendibles por el usuario, pero si la propuesta está basada en una notación formal, está será más difícilmente entendible.

2.1 Incorporación de la usabilidad en la arquitectura del sistema

Son varios los autores de la comunidad de la Ingeniería del Software que han desarrollado aproximaciones para tratar la usabilidad desde las primeras etapas del proceso de desarrollo software con el fin de incorporar la usabilidad en la arquitectura de los sistemas. Estos autores han elaborado propuestas para incorporar aspectos de la comunidad IPO dentro del proceso de desarrollo software. A continuación se presentan algunos de los trabajos más relevantes en orden cronológico.

2.1.1 Definición de la declaración de la usabilidad del sistema

2.1.1.1 Descripción

El trabajo de Comstock [Comstock, 1996] tiene como objetivo el tratar la usabilidad dentro de la arquitectura software del sistema. El propósito inicial era el de encapsular los principios de diseño dentro de un modelo conceptual de manera que permitiera incluir aspectos de usabilidad. Este trabajo está realizado en un ámbito industrial, con los productos software de la empresa *Network Integration Software (NIS)*, empresa que se dedica al desarrollo de aplicaciones que trabajan en un entorno de redes. Estudiando estas aplicaciones, la autora detectó que la usabilidad del sistema dependía de la arquitectura sobre la que estuviera construido ese sistema. Para determinar cómo incluir la usabilidad dentro de la arquitectura, se creó un equipo de analistas expertos en temas de usabilidad, que estarían presentes durante todo el proceso de desarrollo de un producto software de la empresa NIS. El objetivo de este equipo era el de asegurar que las aplicaciones desarrolladas fueran usables. Las pautas seguidas por este equipo para introducir la usabilidad en la arquitectura fueron las siguientes:

1. **Determinar el objetivo de la arquitectura de usabilidad:** Este objetivo debe estar basado en las tareas que realizará el usuario con el sistema.

2. **Supuestos de los analistas:** El equipo de analistas debe entender los requisitos de usabilidad como si fueran sus propios requisitos.
3. **Introducirse en la información del usuario:** El equipo de analistas debe estudiar en profundidad toda la información que posee el usuario referente al sistema a desarrollar.
4. **Temas:** Se deben identificar los temas o aspectos que son cruciales para el usuario y entenderlos como si fueran del propio analista.
5. **Metáforas:** Cada miembro del equipo de analistas debe elegir una metáfora para el sistema de forma que represente significativamente los temas del usuario.
6. **Esencia:** Cada una de las metáforas captura una esencia del sistema. El equipo de analista debe detectar las esencias del sistema a través de las metáforas. Las esencias se pueden ver como las características que debe tener un sistema software ideal.

El equipo de analistas expertos fue siguiendo estas pautas para el desarrollo del sistema, pero no se conseguía llegar a un modelo conceptual que representara el sistema. A raíz de este problema, el equipo de analistas elaboró un informe denominado *la declaración de la usabilidad del sistema*. Este informe recoge una serie de opiniones del equipo de analistas para conseguir maximizar la usabilidad de los sistemas. Esta declaración está formada por seis puntos que se deben seguir para conseguir sistemas usables:

- Se deben integrar las tareas del usuario y sus intenciones sin prestar atención a la tecnología sobre la que se implemente el sistema.
- Se debe reconocer la primacía del usuario.
- Se deben facilitar operaciones en grupo, de forma que un grupo de usuarios se trate como una entidad.

- El cuarto punto está relacionado con el primero y afirma que tomar una aproximación basada en tareas no requiere ningún conocimiento de la tecnología sobre la que se implementen estas tareas.
- Muchos de los sistemas desarrollados obvian la necesidad de futuras modificaciones, es decir, normalmente se asume que los sistemas se van a mantener invariantes para siempre. En cambio, la realidad ha demostrado que con el tiempo todos los sistemas son dinámicos, ya que exigen modificar parte de su comportamiento. Este punto afirma que los sistemas se deben desarrollar teniendo en cuenta que el cambio es parte de sus requisitos.
- El último punto reconoce la importancia de las componentes de la interfaz de usuario y propone el uso de guías de estilo para definir las.

La definición de esta declaración tuvo un gran impacto en el desarrollo de productos por parte de la empresa NIS, quedando empíricamente probada la validez de los puntos que componen la declaración. En cambio, esta declaración no ayudó a definir un modelo conceptual a partir del cual representar la usabilidad de forma abstracta, ya que hubo problemas a la hora de definir algunos de los aspectos de usabilidad.

2.1.1.2 Ventajas y limitaciones de la propuesta

La propuesta de Comstock está pensada para métodos de desarrollo basados en modelos conceptuales. El punto más fuerte de su trabajo es la definición de una serie de buenas prácticas para diseñar sistemas.

Como inconvenientes presenta los siguientes: No especifica para qué tipo de plataforma serían útiles sus propuestas; no incluye una notación para representar la usabilidad; no especifica una notación para representar la usabilidad; no especifica cómo implementar la propuesta en un lenguaje de programación; no especifica los atributos de usabilidad que se tratan en su propuesta; la representación de usabilidad es demasiado compleja como para que el usuario la pueda entender y participar en el proceso de desarrollo.

El objetivo inicial de la autora era el de crear un modelo conceptual que representara la usabilidad. En cambio, debido a la imposibilidad de definir este modelo conceptual, el trabajo se centró en la definición de la declaración de usabilidad del sistema. La autora atribuye este fracaso a la falta de herramientas o de ejemplos que representaran los cambios que se deben llevar a cabo en la arquitectura de los sistemas para incorporar la usabilidad. Sin embargo, la declaración propuesta por la autora tampoco propone nada en ese sentido, es simplemente una serie de buenas prácticas que se deben tener en cuenta para construir sistemas usables (muchos de los cuales ya han sido recomendados antes por la IPO), pero en las cuales no se profundiza.

2.1.2 Relación de usabilidad con la arquitectura a través de escenarios

2.1.2.1 Descripción

Otro autor que resalta la importancia de relacionar la usabilidad con la arquitectura de los sistemas es Bass [Bass, 2003]. Este autor, afirma que la usabilidad es un atributo de calidad lo suficientemente importante como para tenerlo en cuenta durante el diseño de la arquitectura. El trabajo de Bass se centra en el concepto de *escenario* de usabilidad. Cada uno de estos escenarios representa un conjunto de mejoras de usabilidad que se pueden incorporar al sistema modificando su arquitectura. El conjunto de escenarios se ha obtenido a partir de la literatura, de discusión con gente del área y a partir de la experiencia personal. Los escenarios presentan las siguientes características:

- Los escenarios son comunes a varios sistemas software. Es decir, no dependen de un dominio o una funcionalidad específica.
- Los escenarios son arquitectónicamente significativos. Es decir, la incorporación de un escenario al sistema implica que la arquitectura de ese sistema se debe modificar.
- Cada escenario presenta una solución distinta que mejora la usabilidad del sistema.

Cada uno de los escenarios lleva asociado un patrón arquitectónico que proporciona una solución para su implementación. Estos escenarios ponen de manifiesto que la relación entre una mejora de la usabilidad y la arquitectura software es más compleja que la simple separación de interfaz y funcionalidad.

Bass define alrededor de veinte escenarios y propone dos tipos de clasificación. En ambos casos se trata de un proceso *bottom-up*:

- Una clasificación basada en los beneficios de usabilidad que puede obtener el usuario si el escenario se implementa correctamente. Esta clasificación es similar a la llevada a cabo por otros investigadores de la comunidad IPO como por ejemplo Nielsen [Nielsen, 1993].
- Una clasificación basada en los patrones arquitectónicos que los representan. Esta clasificación depende de las implicaciones en la arquitectura que tenga cada uno de los escenarios.

A partir de estas dos clasificaciones, Bass propone una matriz (Figura 2.1) de dos dimensiones que engloba a ambas. Esta matriz la debe usar el equipo de diseñadores para evaluar y diseñar la arquitectura del sistema dependiendo de las exigencias del usuario. La matriz se puede usar de tres formas distintas que no tienen porque ser excluyentes entre sí:

1. El equipo de diseñadores decide qué escenarios son importantes para alcanzar los objetivos de diseño del producto a desarrollar y los buscan en la matriz. A partir de la posición que ocupa cada uno de los escenarios en la matriz, los diseñadores pueden deducir los beneficios que repercutirán al usuario y las implicaciones que tiene en la arquitectura su incorporación al sistema.
2. El equipo de diseñadores identifica los beneficios que son más importantes para el sistema que se debe implementar y en base a ellos identifica las implicaciones que éstos tienen sobre la arquitectura del sistema.
3. En el caso de que se haya propuesto una arquitectura de antemano, los analistas pueden visualizar la línea de la matriz equivalente a

esa arquitectura y descubrir los escenarios que se pueden incorporar a partir de ella para mejorar la usabilidad.

Usability Benefits →		Increases individual effectiveness						Reduces impact of system errors		Increases confidence and comfort	
		Expedites routine performance		Improves non-routine performance		Reduces impact of mistakes		Tolerates system errors	Prevents system errors		
		Accelerates error-free portion	Reduces impact of slips	Supports problem-solving	Facilitates learning	Prevents mistakes	Accommodates mistakes				
Architectural Tactics ↓	Separation	Encapsulation of function	4, 13, 14, 15, 20, 23		4, 13, 20	4, 13, 20	4, 13, 20	9, 14			
	Data from the view of that data	12, 13, 24, 25	12	12, 13, 22, 24, 25, 26	12, 13, 24	12, 13, 22, 24	12			12	
	Data from commands	1, 24, 25	5, 17	5, 17, 24, 25, 26	5, 17, 24	1, 5, 17, 24	1, 5, 17			17	
	Authoring from execution	1, 2	2			1, 2	1, 2				
	Replication	Data	16								
	Commands	2	2	22		2, 22	2				
	Indirection	Data	7, 11, 14	11	7, 11		14				
	Function	6, 14, 20, 27	27	6, 20	20	20, 27	14		6	27	
	Recording	2, 7	2, 3, 21	3, 7, 21		2	2, 3, 21	3, 8			
	Preemptive Scheduling	15, 18, 19	3, 5, 17, 18	3, 5, 10, 17	5, 10, 17	5, 17, 19	3, 5, 17	3		17, 18	
	Models	Task	18, 19	5, 17, 18	5, 10, 17	5, 10, 17	5, 17, 19	5, 17			17, 18
	User	12, 18	5, 12, 17, 18	5, 10, 12, 17, 22	5, 10, 12, 17	5, 12, 17, 22	5, 12, 17			12, 17, 18	
	System	4, 6, 19, 23	3, 5, 17	3, 4, 5, 6, 17	4, 5, 17	4, 5, 17, 19	3, 5, 17	3	6, 23	17	

KEY

- | | | |
|-----------------------------------|------------------------------------|--|
| 1 Aggregating data | 10 Providing good help | 19 Predicting task duration |
| 2 Aggregating commands | 11 Reusing information | 20 Supporting comprehensive searching |
| 3 Canceling commands | 12 Supporting international use | 21 Supporting undo |
| 4 Using applications concurrently | 13 Leveraging human knowledge | 22 Working in an unfamiliar context |
| 5 Checking for correctness | 14 Modifying interfaces | 23 Verifying resources |
| 6 Maintaining device independence | 15 Supporting multiple activities | 24 Operating consistently across views |
| 7 Evaluating the system | 16 Navigating within a single view | 25 Making views accessible |
| 8 Recovering from failure | 17 Observing system state | 26 Supporting visualization |
| 9 Retrieving forgotten passwords | 18 Working at the user's pace | 27 Supporting personalization |

Figura 2.1 Matriz que relaciona los beneficios en la usabilidad con los cambios que hay que añadir en la arquitectura del sistema en 27 escenarios [Bass, 2003]

Un aspecto importante a la hora de diseñar la matriz es la densidad. Si su densidad es baja, cualquier beneficio sobre la usabilidad implicaría muchos cambios en la arquitectura del sistema. En cambio si la densidad es alta, con una única arquitectura muy compleja se resolverían todos los problemas de usabilidad. Según Bass, para conseguir una densidad óptima se deben tomar entre 1 y 5 escenarios por celda de la matriz.

2.1.2.2 Ventajas y limitaciones de la propuesta

La propuesta de Bass está pensada para un desarrollo de sistemas manual. La principal ventaja de su propuesta es la definición de una matriz que relaciona beneficios de usabilidad con la arquitectura del sistema. Esta matriz es una especie de guía para el analista. A pesar de la existencia de esta matriz, la decisión de aplicar o no un escenario a partir de la matriz la deben tomar el analista, y aunque éste disponga de información proporcionada por la matriz, la decisión puede estar influenciada por aspectos subjetivos del propio analista. Esta aproximación tropieza con un inconveniente tradicional de la incorporación de la usabilidad a los sistemas: la visión de usabilidad que se incorpora es la del diseñador y no la del usuario. Esta subjetividad se podría resolver con guías o consejos sobre qué escenarios son más convenientes para determinados tipos de sistemas o determinadas soluciones de usabilidad.

Las principales limitaciones del trabajo de Bass son las siguientes: No especifica para qué tecnología se podría instanciar la propuesta (un mismo escenario puede ser más o menos complejo de incorporar a la arquitectura dependiendo de varios factores como el lenguaje de programación o plataforma); Carece de una notación precisa para representar la usabilidad; No especifica cómo implementar la propuesta en un lenguaje de programación; sí que especifica los atributos de usabilidad que se tratan en su propuesta, pero la cantidad de estos atributos es escasa en comparación con los atributos definidos en la ISO/IEC 9126-1 [ISO/IEC 9126-1, 2001]; la representación de la usabilidad no es entendible por el usuario y por tanto, éste no podrá participar en el proceso de desarrollo.

2.1.3 Framework de usabilidad definido por Folmer

2.1.3.1 Descripción

De entre todos los autores que proponen incorporar la usabilidad en el diseño de la arquitectura, uno de los más relevantes ha sido Folmer [Folmer, 2004]. En su trabajo, Folmer propone un Framework de usabilidad para relacionar los atributos de usabilidad con la arquitectura del sistema. La Figura 2.2

muestra las distintas partes de este Framework, que se dividen en dos grandes áreas, el espacio del problema y el espacio de la solución:

- El espacio del problema trata de definir lo que se entiende por usabilidad, qué atributos la componen y cómo se puede asegurar dentro de un sistema.
- El espacio de la solución proporciona una serie de prácticas y experiencias para mejorar la usabilidad en base a heurísticas, guías y patrones de usabilidad que existen en la literatura. En el espacio de la solución es donde se diseña la usabilidad.

Para relacionar el espacio del problema con el espacio de la solución y por lo tanto relacionar la usabilidad con la arquitectura software, el Framework consta de las siguientes capas, tal y como se distingue en la Figura 2.2:

- **Definición de la usabilidad y de sus atributos:** Esta capa consiste en la definición del término usabilidad y su definición en componentes medibles. Además, alberga las definiciones clásicas, como por ejemplo las proporcionadas por las distintas ISO (como la ISO 9126-1 [ISO/IEC 9126-1, 2001] y la ISO 9241-11 [ISO/IEC 9241-11, 1998]) y autores como Nielsen [Nielsen, 1993] o Shackel [Shackel, 1991].
- **Indicadores de usabilidad:** Los atributos de usabilidad son conceptos abstractos que no son medibles. El objetivo de esta capa es el de proporcionar métricas para estos atributos. Para ello, esta capa está formada por indicadores de usabilidad medibles que están relacionados con los atributos de usabilidad de la capa superior. Por ejemplo, el tiempo de aprendizaje es un indicador medible para el atributo llamado facilidad de aprendizaje (*learnability*).
- **Propiedades de usabilidad:** Esta es la capa que relaciona el espacio del problema con el espacio de la solución a través de las propiedades que la componen. Las propiedades de esta capa se derivan de heurísticos y principios de diseño que sugieren cómo abordar el diseño. Estas propiedades están relacionadas con las decisiones de diseño software, pero no tienen por qué tener una

relación unívoca con los indicadores de usabilidad. Por ejemplo, el patrón que representa el uso de un asistente decreta el tiempo de aprendizaje pero a su vez aumenta el tiempo necesario para realizar la tarea asociada al asistente.

- **Conocimiento del diseño:** Esta capa está formada por heurísticas de diseño, estándares de interfaz y técnicas de diseño, como el modelado centrado en el usuario, modelado de tareas y creación de prototipos. El conocimiento del diseño proporciona directivas de diseño que pueden llegar a ser muy detalladas.

En resumen, el Framework propuesto por Folmer ilustra la relación entre usabilidad y arquitectura software mediante la definición de términos intermedios entre las distintas capas del Framework.

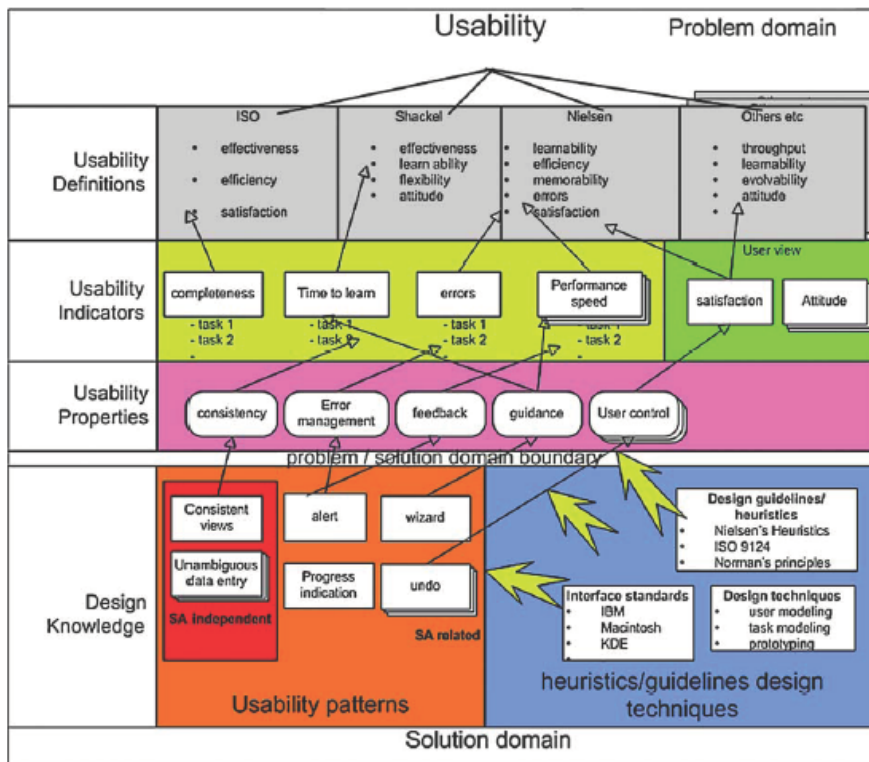


Figura 2.2 Framework de usabilidad propuesto por Folmer [Folmer, 2004]

2.1.3.2 Ventajas y limitaciones de la propuesta

La propuesta de Folmer está orientada a métodos de desarrollo manual. Su principal ventaja es que propone la incorporación de la usabilidad desde las primeras fases del proceso de desarrollo, evitando cambios en la arquitectura a posteriori.

Una de las limitaciones de la propuesta de Folmer es que no especifica la tecnología sobre la cual se podría aplicar la propuesta. Otra de las limitaciones es que no especifica los pasos a seguir para aplicar su propuesta. No se indica de forma precisa cómo operan las relaciones entre capas. Las relaciones definidas son solo relaciones potenciales (posibles). Es el analista que diseñe la arquitectura el que debe establecer estas relaciones según su propio criterio. Además, no se proporciona una notación para representar la usabilidad en las distintas capas.

Otra de las limitaciones es que Folmer presenta la estructura que se debe seguir en los sistemas para implantar la usabilidad a nivel de arquitectura, pero no profundiza sobre cuáles son los elementos que componen cada una de las capas del Framework, sólo muestra unos pocos a modo de ejemplo. Esto hace que su implementación en un lenguaje de programación resulte compleja. Sería necesario llevar a cabo un estudio más detallado para definir los elementos que deben formar parte de cada una de las capas del Framework de modo que éstos queden definidos completamente. Además, también dificulta la implementación de la propuesta el hecho de que no se identifican las implicaciones que puede tener en la arquitectura la mejora de la usabilidad del sistema. Esto se podría solucionar definiendo patrones que englobaran las mejoras de usabilidad que se incorporan a la arquitectura. Estos patrones darían una idea clara sobre cómo incorporar estas mejoras.

Por último, un punto débil de la propuesta de Folmer es que los modelos de usabilidad que propone no son comprensibles por el usuario, y por tanto, no se podrán evaluar hasta ser implementados.

2.1.4 Conclusiones

De todos los trabajos comentados en este apartado, cabe destacar la importancia que dan todos los autores a la necesidad de tener en cuenta la usabilidad desde las primeras fases del desarrollo software en las que se

define la arquitectura del sistema. Si la usabilidad no se tiene en cuenta hasta que el sistema se implementa y se realizan los tests con el usuario, la incorporación de ciertos atributos de usabilidad puede implicar cambios importantes en la arquitectura del sistema. Por lo tanto, el tener en cuenta la usabilidad desde las primeras fases de desarrollo, evita tiempo en el desarrollo del sistema, ya que minimiza las modificaciones de la arquitectura. En esta tesis se tiene en cuenta la propuesta de tratar la usabilidad en la construcción de la arquitectura y para ello se trata la usabilidad desde la fase de modelado conceptual.

Estos trabajos reflejan que la incorporación de la usabilidad a los sistemas no sólo afecta a las interfaces, sino que tiene implicaciones en el diseño del sistema. Por lo tanto, la usabilidad debería guiar el diseño del sistema de la misma forma que el resto de requisitos funcionales.

El inconveniente que comparten todos los autores es que, aunque la arquitectura ya incluya características de usabilidad y esté bien definida, la persona que implemente el sistema puede cometer errores al plasmar dicha arquitectura en el código y por tanto generar aplicaciones que no sean usables. Estos errores sólo implican cambios en la fase de implementación.

2.2 Incorporación de la usabilidad a nivel de requisitos

Algunos autores de la comunidad de la Ingeniería del Software proponen la incorporación de la usabilidad incluso antes del diseño de la arquitectura del sistema, es decir, en la fase de captura de requisitos. Según estos autores, el trato explícito de la usabilidad en la especificación de los requisitos es importante para proporcionar una visibilidad de las características de la usabilidad tanto para los desarrolladores como para los testadores. A continuación se detallan algunos de los trabajos más relevantes que proponen la incorporación de la usabilidad a nivel de captura de requisitos ordenados cronológicamente.

2.2.1 Estilos para la captura de requisitos propuestos por Lauesen

2.2.1.1 Descripción

Existen autores que afirman que no hay ningún estilo mejor que otro para la captura de requisitos de usabilidad. Lo más adecuado es tener una lista de estilos y escoger el que sea más adecuado para un sistema en concreto. En este contexto, se entiende por estilo al conjunto de requisitos que persiguen un mismo objetivo. Los estilos actúan para el analista como guías para la captura de requisitos, permitiéndole elegir los estilos a aplicar dependiendo del sistema a desarrollar. En esta línea se encuentra el trabajo de Lauesen [Lauesen, 1998], el cual propone seis estilos basados en su experiencia y en las propuestas de la comunidad IPO. Los estilos también especifican y miden los factores de usabilidad de forma más o menos directa. Concretamente, los estilos que propone Lauesen son:

1. **Estilo de representación:** En este estilo se especifica cómo de rápido los usuarios pueden aprender varias tareas, cómo de rápido pueden realizarlas tras un entrenamiento, etc. Un ejemplo de este estilo sería, *“los usuarios novatos podrán completar las tareas Q y R en 15 minutos”*. Estos requisitos se pueden especificar a través de tests de usabilidad. Por medio de prototipos, los tests se pueden realizar en las fases tempranas del desarrollo. Uno de los inconvenientes de este estilo es que algunas de las características de la usabilidad como *eficiencia en el uso diario* no se puede estimar en el proceso de desarrollo aun cuando el software esté disponible. El principal problema de este estilo es que se necesita esfuerzo y experiencia para elegir las tareas e iterativamente adaptar el diseño para hacerlo coincidir con la especificación.
2. **Estilo de defecto:** Este estilo se asemeja al anterior, pero en vez de medir el tiempo en realizar una tarea, identifica los defectos de usabilidad y especifica cómo de frecuentemente pueden éstos aparecer. Un defecto de usabilidad es algo que causa que el usuario cometa errores o se sienta molesto. Un ejemplo de este estilo sería: *“en promedio, un usuario novato encontraría menos de*

0,2 problemas de usabilidad serios al realizar las tareas Q y R". El usuario debe pensar en voz alta durante el tests de usabilidad y un observador guarda todos los defectos que encuentre el usuario. La principal ventaja de este estilo es que la lista de defectos proporciona una retroalimentación muy buena para los analistas, permitiéndoles corregir los defectos. La desventaja es que no se puede asegurar que se está capturando la esencia de la usabilidad. Por ejemplo, baja eficiencia en el uso diario se mostrará únicamente como un defecto de usabilidad si el usuario se queja de él, aunque puede que el origen del defecto sea otro distinto a la usabilidad.

- 3. Estilo de proceso:** Este estilo especifica el proceso de desarrollo a utilizar para asegurar la usabilidad. El estilo no dice nada sobre el resultado del desarrollo, pero se espera que este proceso genere un buen resultado. Un ejemplo de este estilo sería: *"durante el diseño, se harán tres prototipos. De cada prototipo se evaluará su usabilidad y se corregirán los defectos detectados"*. Se pueden especificar varios procesos como la evaluación heurística o el diseño de diálogo estructurado. Además, se puede especificar el criterio de terminación para las iteraciones de diseño, por ejemplo, continuar hasta que no se detecten defectos de usabilidad serios. La principal ventaja de este estilo es que evita la necesidad de encontrar valores objetivo, como el tiempo que se tarda en realizar una tarea. La desventaja es que se deja mucho criterio de decisión a los diseñadores. Los diseñadores a menudo seleccionan tareas y usuarios erróneos o sólo hacen pequeñas modificaciones en los prototipos.
- 4. Estilo subjetivo:** En este estilo se pregunta a los usuarios sobre sus opiniones. Normalmente esta consulta se realiza con un cuestionario usando la escala Likert. Un ejemplo de este estilo sería: *"el 80% de los usuarios encontrará el sistema fácil de aprender y eficiente para el uso diario"*. Algunos especialistas aseguran que con este mecanismo se consigue capturar la esencia de la usabilidad. Como inconveniente de este estilo, cabe resaltar que la satisfacción del usuario está influenciada por factores organizacionales fuera del alcance del sistema. Otro problema de este estilo es la dificultad de verificar los requisitos durante el

desarrollo del sistema. Muchos expertos en usabilidad obtienen las opiniones subjetivas de los usuarios después de los tests basados en prototipos, pero estas opiniones no se corresponden con las opiniones de los usuarios una vez el sistema se haya desarrollado.

5. **Estilo de diseño:** Este estilo recomienda los detalles de la interfaz de usuario, principalmente transformando los requisitos de usabilidad en requisitos funcionales. Un ejemplo de este estilo sería: *“el sistema usará las imágenes especificadas en el anexo I”*. Estos requisitos son fáciles de verificar en el producto final y fáciles de seguir durante el proceso de desarrollo. Cuando el ingeniero de requisitos ha hecho un buen trabajo en las tareas de análisis mediante prototipos y tests de usabilidad, los resultados de usabilidad con este estilo son buenos. Sin embargo, normalmente este estilo se aplica sin tests de usabilidad y, por tanto, los resultados son como si la usabilidad no se hubiera tenido en cuenta. Los prototipos que no se evalúan se pueden usar como ejemplos de lo que el usuario tiene en mente, pero no como requisitos de usabilidad.

6. **Estilo de guía:** Son guías de estilo oficiales de facto, o bien guías de la compañía para la que se desarrolla, o basadas en la experiencia. Un ejemplo de este estilo sería: *“el sistema seguirá la guía de estilos de Windows. Los menús no tendrán más de tres niveles de anidamiento”*. Aunque normalmente las guías mejoran la usabilidad, éstas tienen poca esencia de la usabilidad. Es decir, se puede tener un sistema que sea muy complicado de utilizar aunque siga todas las guías de usabilidad.

Independientemente del estilo seleccionado para la captura de requisitos, Lauesen propone una elicitación de requisitos basada en tres pasos:

- **Identificar los aspectos de usabilidad claves en las tareas críticas:** En un sistema complejo, el número de tareas es muy grande y no se puede cubrir el estudio de la usabilidad de todas las tareas. Por lo tanto, es necesario identificar las tareas críticas para centrar los esfuerzos del analista en mejorar la usabilidad de estas tareas.

- **Elegir los estilos de requisitos:** No se debe usar el mismo estilo para capturar todos los requisitos de usabilidad. Dependiendo del requisito, hay unos estilos más adecuados que otros. El autor ha propuesto una tabla a partir de la cual, dependiendo de la característica de usabilidad, el analista debe utilizar un estilo u otro (Figura 2.3).

Issue	Style					
	Performance	Defect	Process	Subjective	Design	Guideline
1. Recording experience data, efficiency	X		X			
2. Marketing, learn on your own, particularly using experience data		X	X			
3. Invoicing, efficiency, understanding, overview	X		X			X
4. Invoicing, scroll and search time	(X)					
5. Accounting, cut-over course	X					
6. Accounting, efficiency	X					
7. Detail planning, learning	X		X			
8. Detail planning, efficiency, overview	X		X			
9. Easy to switch between systems						X

Figura 2.3 Tabla con los estilos de requisitos por cada característica de usabilidad [Lauesen, 1998]

- **Elegir métricas y valores objetivo:** Se deben definir métricas a partir de las cuales verificar el nivel de usabilidad.

Los estilos y la elicitación de requisitos propuestos por Lauesen están empíricamente validados mediante el desarrollo de un sistema de gestión de unos astilleros.

2.2.1.2 Ventajas y limitaciones de la propuesta

La propuesta de Lauesen está orientada a métodos de desarrollo manuales. Una de las principales ventajas de esa propuesta es que especifica qué

características de usabilidad están en cada estilo. El total de estilos propuestos por Lauesen son seis, y dependiendo de las características de usabilidad que se consideran más importantes para un sistema en concreto se deben elegir unos estilos u otros. El autor propone una tabla (Figura 2.3) en la que clasifica los estilos por características de usabilidad. Otra de las ventajas de la propuesta es que especifica los atributos de usabilidad que trata, aunque el número de atributos tratados no es muy elevado (nueve atributos). Otra de las mejoras es que el usuario puede participar en el proceso de desarrollo, ya que Lauesen propone el uso de prototipos para validar las interfaces con los usuarios.

En cuanto a los inconvenientes cabe destacar los siguientes: no especifica en qué plataforma o tecnología se podría usar la propuesta; no especifica una notación para representar la usabilidad; no especifica cómo implementar la propuesta en un lenguaje de programación.

2.2.2 Propuesta para incorporar requisitos de calidad al proceso de desarrollo

2.2.2.1 Descripción

Algunos autores como Bevan [Bevan, 2000] han centrado su trabajo en la incorporación de la calidad en uso dentro del proceso de desarrollo software a partir de la especificación de los requisitos. Tal y como está definido en la ISO/IEC 9126-1 [ISO/IEC 9126-1, 2001], la calidad en uso es la visión que tienen los usuarios de la calidad del sistema y está medida en términos del resultado de la utilización del sistema, más que de las propiedades del propio sistema.

En concreto, Bevan participó en el proyecto TRUMP, cuyo objetivo era incrementar la calidad de los sistemas introduciendo métodos para tratar la usabilidad dentro del proceso de desarrollo software y promocionar la conciencia de usabilidad dentro de la cultura de la organización. El proyecto TRUMP combina tres aproximaciones complementarias para mejorar la calidad del sistema desde una perspectiva basada en el usuario:

- Mejorar la calidad del proceso de desarrollo software incorporando actividades centradas en el usuario derivadas de la ISO 13407 [ISO

13407, 2001] y del Modelo de Madurez de la Usabilidad de la ISO TR 18529 [ISO TR 18529, 2000].

- Mejorar la calidad del sistema por medio de la mejora de la calidad de la interfaz de usuario.
- Mejorar la calidad en uso asegurando que el sistema coincide con las necesidades del usuario en base a efectividad en uso, productividad y satisfacción.

Los pasos que componen el método desarrollado en el proyecto TRUMP son:

- 1. Reunión con los stakeholders:** En este paso se identifican los roles de la usabilidad junto con el contexto de uso y los objetivos de usabilidad, y cómo éstos se relacionan con los objetivos de la lógica de negocio.
- 2. Contexto de uso:** Se debe recoger la información referente a los usuarios, sus tareas, y las restricciones técnicas y ambientales.
- 3. Escenarios de uso:** Se documentan ejemplos sobre cómo los usuarios esperan desarrollar las tareas clave en un contexto específico que proporcione una entrada para el diseño y una base para los subsiguientes tests de usabilidad.
- 4. Evaluar un sistema existente:** Se evalúa una versión anterior o un sistema competidor con el propósito de identificar problemas de usabilidad y obtener medidas de la usabilidad como entrada para los requisitos de usabilidad.
- 5. Requisitos de usabilidad:** Se deben establecer los requisitos de usabilidad para los grupos de usuarios y tareas detectadas en el contexto de uso y en los escenarios.
- 6. Prototipado en papel:** Se evalúa con los usuarios unos prototipos que se construyen de forma rápida utilizando bocetos en papel. Mediante estos bocetos se clarifican los requisitos y se permite una comprobación rápida del diseño de la interfaz.

- 7. Guía de estilo:** El diseño del sistema debe seguir las guías de estilo de la empresa para la que se desarrolla el sistema.
- 8. Evaluación de prototipos máquina:** Se debe realizar una comprobación informal usando entre tres y cinco usuarios con las tareas clave. De esta forma, se proporciona una retroalimentación rápida de los prototipos de usabilidad.
- 9. Evaluación de la usabilidad:** Se debe realizar una evaluación formal de la usabilidad con ocho miembros representativos de un grupo de usuarios desarrollando tareas clave. Así, se pueden identificar los problemas de usabilidad que aún existan y evaluar si los objetivos de usabilidad se han alcanzado.
- 10. Recoger retroalimentación de los usuarios:** Se debe recoger información de recursos como cuestionarios de usabilidad, líneas de ayuda y servicios de soporte para identificar cualquier problema que se debería resolver en futuras versiones.

Este método desarrollado por Bevan está empíricamente validado ya que se aplicó para el desarrollo de dos sistemas reales. Por un lado, se aplicó en el desarrollo de un sistema para la agencia tributaria del Reino Unido. Esta aplicación debía soportar una plantilla de más de 60.000 personas en más de 600 oficinas. Por otro lado, se aplicó en el desarrollo de un sistema para la industria de aviación de Israel, en la que trabajan unas 100 personas.

Los resultados del experimento en ambos sistemas fue similar: los usuarios encontraban ambas aplicaciones fáciles de aprender y sentían que tenían el control sobre el sistema en todo momento. Además, los resultados mostraban que los usuarios no percibieron el sistema como un sistema que estuviera en todo momento proporcionando ayuda al usuario ni tampoco lo encontraron muy eficiente.

2.2.2.2 Ventajas y limitaciones de la propuesta

La propuesta de Bevan está orientada a métodos de desarrollo manuales. Como principal ventaja, el usuario forma parte esencial en muchas de sus fases del proceso de desarrollo. Esto hace posible que los requisitos de calidad representen fielmente los deseos del usuario. Otra de las ventajas

es el uso de prototipos para evaluar la usabilidad con el usuario. El uso de prototipos facilita la evaluación del usuario, el cual no tiene que conocer aspectos técnicos para participar en el proceso de desarrollo. Por último, la propuesta de Bevan tiene como ventaja que define una serie de pasos para aplicar la propuesta. Estos pasos son una guía para el analista.

En cuanto a los inconvenientes, el más importante es que aunque existen pasos para la propuesta, no se ha definido una notación precisa para representar la usabilidad en cada uno de los pasos. Por ejemplo, el paso 2 llamado *Contexto de uso* incluye la captura de las tareas que desarrollará el usuario con el sistema, pero no especifica cómo representar estas tareas. Hay muchas notaciones para representar las tareas de usuario, como por ejemplo, CTT [Paternò, 2004] o Diagramas de Secuencia [UML, 2008]). Los autores no indican cuál puede ser la más conveniente o cuál era la que utilizaron en sus evaluaciones empíricas.

Además, se han encontrado los siguientes inconvenientes: no se especifica para qué plataforma o tecnología se podría aplicar la propuesta; no se especifica cómo implementar la propuesta en un lenguaje de programación; no se especifican los atributos de usabilidad que se tratan en la propuesta.

2.2.3 Captura de los requisitos de usabilidad mediante i^*

2.2.3.1 Descripción

Las notaciones utilizadas en la captura de requisitos son muy variadas en la literatura. No obstante, la inmensa mayoría de las notaciones provienen de la comunidad IPO. En cambio, hay algunos autores que han propuesto utilizar notaciones de modelado conceptual propios de la ingeniería del software. Dentro de este último grupo está el trabajo de Cysneiros [Cysneiros, 2003], quien propone la notación i^* [Yu, 1997] para la captura de requisitos no funcionales y específicamente, para los requisitos de usabilidad. Usando la notación i^* , Cysneiros propone la definición de un catálogo usado para guiar a los analistas en la captura de requisitos a través de las distintas posibilidades de usabilidad. El objetivo del modelo i^* es en este caso el modelado de la usabilidad.

Esta forma de modelar los requisitos de la usabilidad da una visión global de todos ellos y permite determinar las relaciones que hay entre los propios requisitos de usabilidad, e incluso entre los requisitos de usabilidad y los requisitos funcionales. Estas relaciones se deben tener en cuenta porque los requisitos de usabilidad no son entes independientes. Es decir, cuando un analista decide alcanzar un objetivo de usabilidad, este objetivo puede entrar en conflicto con otros requisitos no funcionales. Por ejemplo, cuando se decide usar dos contraseñas distintas para entrar al sistema se aumenta la seguridad, pero se disminuye la usabilidad.

Cysneiros ha utilizado la notación i^* porque además de permitir visualizar las relaciones entre los requisitos, el autor quiere usar una notación orientada a agentes. Según Cysneiros, las abstracciones de los agentes permiten ocultar los detalles de las acciones dentro de los criterios de los agentes. Además, el modelado orientado a agentes ofrece un alto nivel de abstracción que es fiel a la representación del mundo real.

En la propuesta de Cysneiros, la usabilidad se va refinando en subobjetivos y en subsubobjetivos, relacionándolos eventualmente con mecanismos implementables. Varios subobjetivos y mecanismos pueden contribuir a variar el grado de usabilidad. Las distintas interpretaciones de usabilidad se pueden recoger y organizar en un catálogo durante la elicitación de requisitos, análisis y diseño. Disponiendo de este catálogo, se puede reutilizar su conocimiento o añadir nuevo. Este conocimiento se representa mediante una estructura en grafo que permite la representación del conocimiento de la organización. El catálogo representa las diferentes alternativas para alcanzar el mismo objetivo, de esta forma el analista puede seleccionar aquella que mejor se adapte al dominio analizado. Además, también se representa si una solución concreta puede contribuir positiva o negativamente con otros requisitos. Los componentes del catálogo se deben descomponer hasta que se pueda afirmar que se ha alcanzado la operacionalización de dichos requisitos no funcionales. La Figura 2.4 muestra parte del catálogo de usabilidad con la notación i^* .

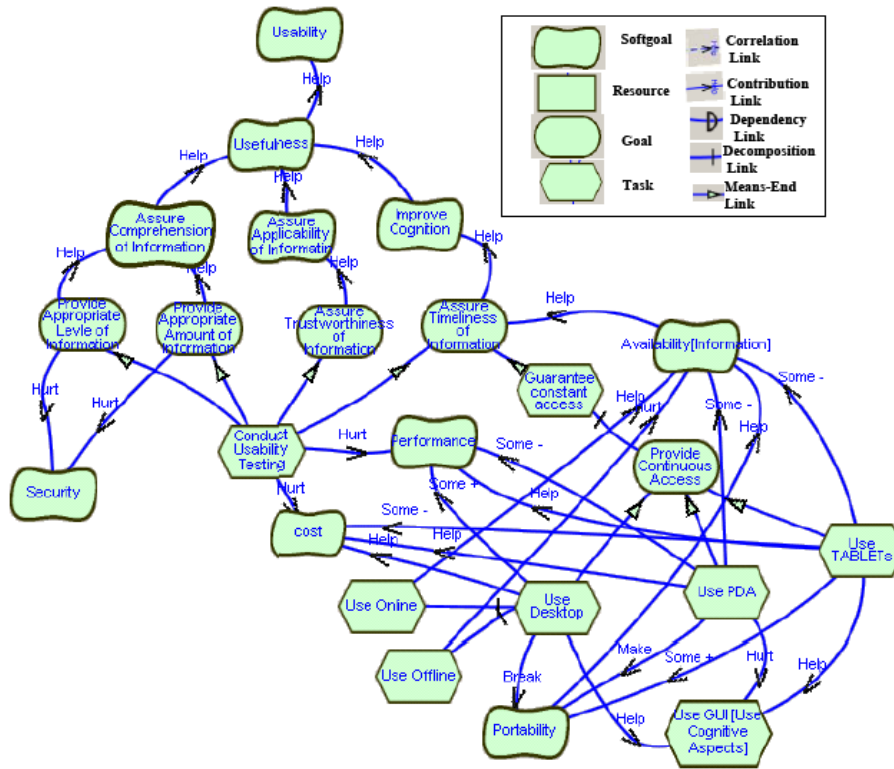


Figura 2.4 Parte del catálogo de usabilidad de Cysneiros [Cysneiros, 2003]

Para elicitar las características de usabilidad se debe consultar con los *stakeholders* qué aspectos de usabilidad corresponden a cada uno de los agentes implicados. Por ejemplo, en el dominio de la medicina, una de las relaciones más importantes se da entre paciente y médico. El paciente espera ser atendido por el médico y el médico espera utilizar un tratamiento en base a medicamentos, ejercicios y una dieta específica. El tratamiento puede implicar el tener que cambiar la medicación o la dieta constantemente. Una forma de ayudar al paciente a seguir el tratamiento es proporcionarle un agente para asegurar que modifica su tratamiento. En base a este agente y a las necesidades del paciente, se determina las características de usabilidad que deben primar, como *utilidad* y *disponibilidad*.

2.2.3.2 Ventajas y limitaciones de la propuesta

La propuesta de Cysneiros está orientada a métodos de desarrollo manuales. La principal ventaja de esta propuesta es el uso de una notación específica, como es i^* .

El uso de i^* añade ciertos problemas a la propuesta de Cysneiros. Aunque en esta notación se pueden representar las relaciones entre los distintos requisitos del sistema y sus agentes, esta notación tiene como principal inconveniente que la semántica de sus elementos no está definida de forma inequívoca y esto puede dar lugar a que un mismo modelo representado con i^* se entienda de forma distinta entre cada uno de los analistas que participan en el desarrollo del sistema. Otros de los inconvenientes que presenta la notación i^* son según [Estrada 2006][Grau, 2006]: la ambigüedad de los modelos; la especificación de los modelos obliga a pensar en fórmulas que se alejan del lenguaje natural; se pueden presentar contradicciones en un mismo modelo.

Otros de los inconvenientes de la propuesta son: no se especifica para qué tipo de plataforma o tecnología se podría utilizar la propuesta; no se definen pasos o guías para el analista; no se especifica cómo implementar la propuesta en un lenguaje de programación; no se mencionan los atributos de usabilidad que se tratan en la propuesta; el nivel de complejidad de i^* hace que un usuario que no conozca esa notación no pueda validar los modelos, y por tanto, no pueda validar la usabilidad del sistema.

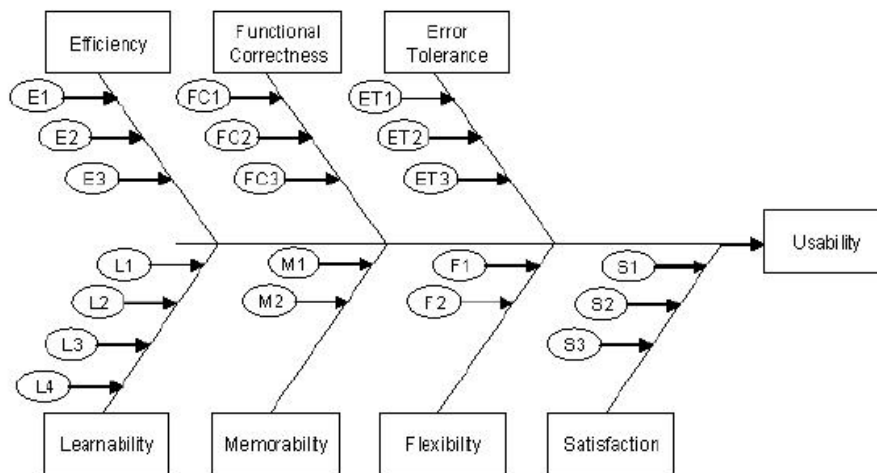
2.2.4 Propuesta de modelado del usuario y de la usabilidad de Adikari

2.2.4.1 Descripción

Adikari [Adikari, 2006] propone incorporar el modelado de los usuarios y de la usabilidad a nivel de captura de requisitos para conseguir mejoras en el diseño del sistema. Esta propuesta está basada en dos modelos: el Modelo de Usuario y el Modelo de Usabilidad. Estos dos modelos se utilizan para comprender y elaborar la especificación funcional de sistemas interactivos.

- **Modelo de Usuarios:** Este modelo es una representación de información y supuestos sobre usuarios que se puede ver desde tres perspectivas distintas:
 - Modelado del conocimiento del usuario: Este modelado implica la estimación precisa del conocimiento de los usuarios, habilidades y experiencias.
 - Modelado de los planes del usuario: Este modelado representa la secuencia de las tareas que los usuarios necesitan realizar para alcanzar sus objetivos.
 - Modelado de las preferencias del usuario: Se centra principalmente en las necesidades de información de los usuarios y sus preferencias.

- **Modelo de Usabilidad:** El Modelo de Usabilidad que propone Adikari está formado por siete atributos: eficiencia del usuario, corrección funcional desde la perspectiva del usuario, tolerancia a errores del usuario, facilidad de aprendizaje, facilidad de memorización, flexibilidad y satisfacción. Para cada uno de estos atributos, Adikari propone varias métricas basadas en aspectos del sistema para calcular su valor de usabilidad. La Figura 2.5 muestra en detalle los componentes de este modelo.



Efficiency	Functional Correctness	Error Tolerance
E1-Task completion in minimum time	FC1-Task completion in minimum time	ET1-Appropriate error messaging for invalid conditions
E2-User tasks are not misleading	FC2-User tasks are appropriate, effective and match the user needs	ET-2 Ability to exit error conditions or unwanted states
E3-No workarounds are needed	FC3-User spends minimal time on "Help"	ET-3 No workarounds are needed
	Satisfaction	
	S1-User desirability of the system and user tasks	
	S2-User opinion about user experience	
	S3-User opinion about frustration or confusion	
	Learnability	
	L1-Clear visibility of current system status and a feel about what to do next	
Memorability	L2-User tasks are not misleading	Flexibility
M1-No memory recall to carry out tasks	L3-Task completion in minimum time	F1-Multiplicity of ways to carry out user tasks
M2-User spends minimal time on "Help"	L4-User spends minimal time on "Help"	F2-User control of task performance

Figura 2.5. Modelo de Usabilidad Propuesto por Adikari [Adikari, 2006]

El método que propone Adikari para incorporar la usabilidad a nivel de requisitos comienza por presentar al diseñador de la interfaz las especificaciones funcionales del sistema, el cual produce el diseño de la interfaz. Posteriormente, se proporcionan al diseñador los Modelos de usuario y de Usabilidad para que refine el diseño de la interfaz. Los pasos que componen este método son:

1. Los diseñadores de la interfaz construyen el Modelo del Usuario y el Modelo de Usabilidad.

2. Los diseñadores de la interfaz diseñan y producen las interfaces basadas únicamente en las especificaciones funcionales del sistema.
3. Los diseñadores de la interfaz rediseñan la interfaz y producen una nueva interfaz a partir de la interfaz generada en el paso 2 en base al Modelo de Usuario y al Modelo de Usabilidad.
4. Los diseñadores de la interfaz rellenan un cuestionario expresando sus vistas, experiencia en el diseño y opiniones sobre el rediseño de la interfaz.
5. Los testeadores de la interfaz llevan a cabo un proceso de evaluación teniendo en cuenta los requisitos del usuario y los requisitos de usabilidad en ambas interfaces (las generadas en el paso 2 y 3) con la ayuda de usuarios.
6. Los resultados de la evaluación se comparan con otras recomendaciones.

El incorporar el Modelo de Usuario y el Modelo de Usabilidad a la especificación funcional hace más detallada la especificación de requisitos. De esta forma se integra el diseño centrado en el usuario al desarrollo de interfaces a través de la especificación de requisitos.

2.2.4.2 Ventajas y limitaciones de la propuesta

La propuesta de Adikari está orientada a métodos de desarrollo manuales. La principal ventaja de esta propuesta es que incluye una serie de pasos para guiar la labor del analista. En cambio, tiene como inconveniente que no especifica la notación a utilizar en cada uno de los pasos. Por ejemplo, no se especifica la notación a utilizar para construir el Modelo de Usuario y el Modelo de Usabilidad. En cuanto a la aplicación de los pasos, se observa que es necesario realizar dos diseños de las interfaces. Por un lado, un diseño teniendo en cuenta sólo los aspectos funcionales del sistema y por otro lado, el diseño teniendo en cuenta el Modelo de Usuario y el Modelo de Usabilidad. El realizar dos diseños aumenta el tiempo de desarrollo de sistemas y puede que todos los recursos empleados en el primer diseño no tengan ningún valor si en el segundo diseño se modifica por completo. Por

tanto, no está justificado que se tengan que hacer dos diseños distintos, parece más eficiente tener en cuenta el Modelo de Usuario y de Usabilidad en el primero de los diseños.

Otros de los inconvenientes son: no se especifica para qué tecnología se puede utilizar la propuesta; no especifica cómo implementar la propuesta en un lenguaje de programación; no especifica los atributos de usabilidad que abarca la propuesta; el usuario no puede participar en el proceso de desarrollo porque son necesarios conocimientos informáticos para realizar las validaciones en base a modelos.

2.2.5 Captura de requisitos de mecanismos de usabilidad

2.2.5.1 Descripción

Otros autores, siguiendo la propuesta de Folmer y Bass, abogan por incorporar la usabilidad en la arquitectura del sistema, ya que la usabilidad no incluye únicamente aspectos visuales sino también funcionales. Para ello, han elaborado un método para la captura de requisitos de usabilidad que guíe la construcción de la arquitectura del sistema. Entre estos autores se encuentran los trabajos de Juristo [Juristo, 2007, b]. Este autor ha identificado un conjunto de características de usabilidad llamadas *Functional Usability Features (FUF)*. Los FUFs son características de usabilidad que afectan no sólo a aspectos visuales de la interfaz, sino también al diseño de la arquitectura del sistema. La lista de FUFs se ha elaborado en base a las características de usabilidad definidas en la literatura y en base a las implicaciones que tienen en el diseño dichas características (de acuerdo con el proyecto STATUS [Juristo, 2003] y Bass [Bass, 2003]). Los FUFs se dividen en subtipos más especializados que se llaman mecanismos de usabilidad.

Los mecanismos de usabilidad se deben incorporar en la arquitectura del sistema desde las primeras fases del desarrollo. Para ello, Juristo ha propuesto una serie de guías para la captura de requisitos que se utilizan durante la elicitación de mecanismos de usabilidad y el proceso de especificación. Su aproximación consiste en definir un conjunto de guías

que faciliten la labor de los analistas en la captura de los requisitos de usabilidad, independientemente de su conocimiento en el área. Estas guías ayudan al analista a entender las implicaciones y conocer cómo elicitar y especificar mecanismos de usabilidad para un sistema software.

Juristo ha analizado la información proporcionada por la comunidad IPO desde una perspectiva de la IS y ha elaborado las guías para la captura de requisitos de usabilidad. Como primer paso, se extrajo y se categorizó la información sobre los mecanismos de usabilidad proporcionada por diferentes autores de la comunidad IPO. Esta información ha servido como base para identificar qué aspectos se deben discutir con los *stakeholders* durante el proceso de elicitación. Sin embargo, no había suficiente información para derivar la especificación y elicitación de todos los mecanismos de usabilidad. Para aquellos mecanismos sin especificación, el autor propone una serie de especificaciones mediante las cuales construir las guías para la captura de requisitos de usabilidad.

Las guías para la captura de requisitos de usabilidad están encapsuladas en patrones de elicitación de usabilidad. La principal ventaja del uso de patrones es el reuso que proporcionan; los patrones que capturan diferentes requisitos se pueden reutilizar en el desarrollo de varios sistemas. Estos patrones ayudan a los desarrolladores a extraer la información necesaria para especificar los mecanismos de usabilidad funcionales. Juristo ha propuesto un patrón por cada uno de los mecanismos de usabilidad definidos.

Los patrones van acompañados de una serie de preguntas que el analista debe formular al usuario para capturar los requisitos de usabilidad. No todas las preguntas que acompañan la captura de requisitos de los patrones tienen el mismo nivel de implicación de los *stakeholders*. Se pueden identificar tres tipos de preguntas:

- Preguntas que el usuario puede responder por sí solo.
- Preguntas que se pueden responder siguiendo recomendaciones del experto en interacción y usabilidad.

- Preguntas que el desarrollador debe responder pero a las cuales el usuario debe dar su opinión.

Las discusiones de estas preguntas deben quedar reflejadas en una tabla, llamada *Tabla de requisitos*. La tabla de requisitos ayuda a los desarrolladores a pensar sistemáticamente en los efectos de los mecanismos de usabilidad a lo largo de todo el proceso de desarrollo.

Las guías para la captura de requisitos de usabilidad se han utilizado en casos de estudio desarrollados por estudiantes de máster. Se han analizado los beneficios potenciales de los patrones de elicitación de usabilidad a diferentes niveles. Para ello se ha trabajado con sistemas en los cuales se han incorporado mecanismos de usabilidad con el objetivo de mejorar su usabilidad. De esta forma se ha podido evaluar el porcentaje de funcionalidad añadida por los mecanismos, los patrones que se han utilizado incorrectamente, y la relación entre los problemas de usabilidad y los patrones.

2.2.5.2 Ventajas y limitaciones de la propuesta

El trabajo de Juristo está pensado para métodos de desarrollo manuales. En cuanto a sus ventajas, cabe destacar que proporciona unos pasos o guías para que el analista capture los requisitos de usabilidad. Por un lado, tiene una metodología clara sobre cómo incorporar sus mecanismos de usabilidad desde las etapas tempranas de construcción del software. Por otro lado, define detalladamente con ejemplos los mecanismos de usabilidad y las guías de captura de requisitos.

Otra de las ventajas es que Juristo propone una notación para representar la usabilidad. Esta notación está basada en el uso de Diagramas de Clase y de Secuencia. Dichos diagramas dan una breve descripción sobre cómo implementar la propuesta en un lenguaje de programación Orientado a Objetos. Además, se detallan los atributos de usabilidad que están incluidos en cada uno de los mecanismos de usabilidad. Por último, también cabe destacar como ventaja que la forma de capturar los requisitos es entendible por los usuarios porque esta captura está basada en el uso de preguntas cercanas a los usuarios. Este hecho facilita la incorporación del usuario en el proceso de desarrollo.

Uno de los inconvenientes de esta propuesta es que el conjunto de mecanismos de usabilidad no es muy amplio y está centrado únicamente en aquellos mecanismos con implicaciones funcionales. Esto puede hacer que algunos de los requisitos de usabilidad exigidos por el usuario no se puedan solucionar con los mecanismos de usabilidad.

Otro de los inconvenientes que se encuentra en la definición de los patrones es que no se especifica sobre qué tipo de sistemas o tecnología se podrían utilizar y en cuáles no.

2.2.6 Conclusiones

Anteriormente se ha dedicado una sección para explicar trabajos que hacían especial hincapié en tratar la usabilidad desde la fase del proceso de desarrollo software en el que se define la arquitectura. En esta sección se han mostrado algunos de los trabajos más relevantes que adelantan el tratamiento de la usabilidad a una fase anterior: la fase de captura de requisitos. Estos requisitos son los encargados de guiar la construcción de la arquitectura.

Este grupo de trabajo es el que más propuestas alberga. Además, cada autor propone modelos distintos para capturar los requisitos de usabilidad, como modelos en i^* , modelos de usuario, estilos, etc. Esto hace que sea un conjunto de trabajos bastante heterogéneo. De todo este grupo de autores, el trabajo que más se aproxima al tipo de solución que estamos buscando son los mecanismos de usabilidad de Juristo [Juristo, 2007, a]. La descripción que hace este autor de los mecanismos hace que se puedan incorporar características de usabilidad no sólo a nivel de requisitos, sino también dentro de la fase de análisis de un proceso de desarrollo MDD. El resto de propuestas son demasiado complejas para tratarlas en un entorno MDD (por ejemplo las de Cysneiros y Adikari) o bien son propuestas muy enfocadas a la implementación del código del sistema (por ejemplo las de Lauesen y Bevan). En cambio, no se ha encontrado ninguna propuesta para la captura de requisitos de usabilidad con el fin de incorporarlas dentro de un proceso MDD. Todas las propuestas están enfocadas desde una perspectiva de desarrollo tradicional.

2.3 Patrones de usabilidad

Esta sección muestra algunos de los trabajos más relevantes relacionados con la definición de patrones de usabilidad por parte de la comunidad IPO. El término de patrón fue definido por Alexander como un método estructurado de diseño para describir una serie de buenas prácticas de diseño en un área particular [Alexander, 1977]. Se caracteriza por:

- Descubrir y nombrar los problemas más comunes en el campo de interés.
- Describir las características principales de las soluciones efectivas para llegar al objetivo marcado.
- Ayudar al diseñador a moverse de un problema a otro de una forma lógica.
- Permitir diferentes caminos en un mismo proceso de diseño.

El concepto de patrón de usabilidad se basa en la definición propuesta por Alexander. Según Perzel [Perzel, 1999], se define el patrón de usabilidad como *la descripción de soluciones que mejoran los atributos de usabilidad*. Los aspectos de usabilidad tratados con estos patrones se refieren básicamente a aspectos de interacción entre el usuario y el sistema. Estos patrones proporcionan soluciones a problemas de usabilidad detectados por la comunidad IPO. Para ello, se centran en la esencia de un problema y su solución, abstrayendo los aspectos del diseño necesarios para aportar las soluciones. A continuación, se presentan algunos de los trabajos de patrones más relevantes ordenados cronológicamente.

2.3.1 Patrones de usabilidad de Perzel para ambientes Web

2.3.1.1 Definición

Uno de los patrones más conocidos son los definidos por Perzel [Perzel, 1999]. Este autor ha descrito una serie de patrones orientados a la Web. En la Web hay problemas de usabilidad que son específicos para este ámbito y que no están presentes en aplicaciones de escritorio. Además, dentro de la Web se pueden distinguir dos tipos de sistemas distintos: las aplicaciones

Web, donde los usuarios deben introducir información; y los sitios Web, donde los usuarios navegan y visualizan información. Perzel propone patrones que afectan a los dos tipos de ambientes Web:

- **La zanahoria y el palo:** Este patrón se usa para conseguir que el usuario proporcione información personal que normalmente es reacio a compartir. Consiste en proporcionar al usuario los estímulos necesarios para incitarlo a introducir dicha información.
- **Política de seguridad:** Este patrón propone cómo dar la suficiente confianza al usuario para que proporcione su información personal al sistema.
- **Marcar los campos obligatorios:** Con este patrón se asegura que el usuario reconoce los campos que se deben rellenar obligatoriamente.
- **Lo que se ve es lo que se puede alcanzar:** Este patrón es una forma de asegurar que el usuario ve todo lo que necesita ver en los contextos del sistema.
- **Plan B:** Con este patrón se ayuda a los usuarios que no pueden percibir correctamente los gráficos del sistema.
- **Validación en el cliente:** Este patrón permite validar la información proporcionada por el usuario antes de enviarla al servidor, es decir, se valida en el propio navegador.
- **Validación en el servidor:** La funcionalidad de este patrón es lo opuesto a la validación anterior. La validación que contiene este patrón se realiza en la parte servidora del sistema software, por lo que el sistema debe enviar la información a validar al servidor.
- **Tres clicks o estás fuera:** Este patrón indica que el usuario debe tener al alcance toda la funcionalidad del sistema con tres clicks de ratón como máximo.

- **Contexto del sitio:** Con este patrón se indica al usuario en qué contexto se encuentra con respecto al resto de contextos del sistema.
- **Localización, localización, localización:** Ubicar las características esperadas por el usuario en la misma posición a lo largo de todos los contextos del sistema.
- **Usted se encuentra aquí:** Proporcionar información al usuario con el objetivo de indicarle en qué contexto se encuentra en todo momento y ofrecerle la capacidad de llegar a este contexto de forma rápida.
- **Colocar etiquetas:** Este patrón marca la profundidad (a nivel de contextos) que se ha alcanzado para llegar al contexto actual.
- **Navegación universal:** Este patrón asegura que se proporcionan al usuario herramientas de navegación en todos los contextos.
- **Buscando en la Web:** Este patrón se usa para proporcionar al usuario buscadores de información.
- **Formularios de registro de aplicaciones Web:** La funcionalidad de este patrón es la de solicitar al usuario que se registre e informarle de las implicaciones que tiene dicho registro.
- **Enlaces y saltos:** Este patrón asegura que cualquier enlace del sistema navega a otro contexto pero dentro de la misma ventana del navegador. Los enlaces que abren nuevas ventanas sólo se deben usar para aquellos casos en los que el usuario debe visualizar el contenido de dos contextos simultáneamente.
- **Formularios de consulta:** Este patrón se utiliza para recuperar información en base a consultas que se pueden lanzar desde la interfaz del contexto.
- **Formularios de entrada de datos:** Este patrón tiene la funcionalidad de minimizar el número de pasos que debe realizar el usuario para rellenar los formularios del sistema.

2.3.1.2 Ventajas y limitaciones de la propuesta

La propuesta de Perzel está orientada a métodos de desarrollo manuales. La principal ventaja de esta propuesta es la existencia de una notación para representar la usabilidad. Concretamente, se utiliza una notación basada en el lenguaje de patrones. Además, también cabe destacar como ventaja que el marco de aplicación de la propuesta de Perzel está bien definido. Concretamente, está pensado para el desarrollo de aplicaciones Web.

En cuanto a los inconvenientes, se puede resaltar que no existe una serie de pasos para guiar la labor del analista, sólo la definición de los patrones. Además, esta definición no es entendible por el usuario porque carece de ejemplos. Por lo tanto, el usuario no podrá participar en el proceso de desarrollo software.

Estos patrones para la Web provienen de propuestas de la IPO, tradicionalmente más preocupados por el usuario, dejando normalmente de lado los temas de implementación. De ahí que la definición de estos patrones no entre en detalle sobre cómo implementarlos en el sistema ni las implicaciones que pueden tener sobre la arquitectura del sistema.

Por último, otro de los inconvenientes es que no se especifican los atributos de usabilidad con los que se relaciona cada uno de los patrones.

2.3.2 Patrones de interacción de Welie

2.3.2.1 Definición

Uno de los trabajos que propone cómo acercar los patrones de usabilidad al usuario final es el llevado a cabo por Welie [Welie, 2000]. Los patrones propuestos en su trabajo están centrados en resolver los problemas de usabilidad que los usuarios tienen con el sistema. Otros trabajos del mismo ámbito como los de Perzel se diferencian de los de Welie en que éste hace una distinción explícita entre la perspectiva del *usuario* y la perspectiva del *diseño*. Al tomar la perspectiva del usuario es importante remarcar el *cómo* y el *por qué* se mejora la usabilidad, mientras que desde la perspectiva del diseño los patrones sólo resuelven problemas que tienen los diseñadores del sistema.

Welie define los patrones como tareas relacionadas que están categorizadas según el tipo de problema de usabilidad que solucionan. Cada uno de los patrones que propone mejora al menos uno de los indicadores de usabilidad siguientes: facilidad de aprendizaje, facilidad de memorización, velocidad de rendimiento, ratio de errores, satisfacción y grado de completitud de tareas.

A continuación se presentan los patrones de usabilidad que define Welie en su trabajo y los problemas que resuelve cada uno de ellos:

- **Asistente:** El usuario quiere alcanzar un objetivo pero se deben tomar varias decisiones para alcanzarlo que puede desconocer.
- **Organización de la información:** El usuario debe entender la información que se muestra por pantalla de forma rápida y tomar medidas dependiendo de esa información.
- **Progreso:** El usuario quiere conocer en todo momento si la operación que solicitó sobre el sistema se está aún ejecutando y además cuanto tiempo resta para su finalización.
- **Escudo:** El usuario puede ejecutar acciones con acciones irreversibles. El sistema mediante este patrón debe ser capaz de avisar al usuario de estas operaciones irreversibles antes de su ejecución.
- **Preferencias:** El usuario puede personalizar el sistema a su gusto.
- **Menú contextual:** En todo momento los usuarios deben saber cuáles son sus alternativas en la ejecución del sistema para saber cuál de todas ellas escoger.
- **Foco:** El usuario debe tener a su disposición información sobre cualquier objeto de la interfaz y cómo llevar a cabo su modificación.
- **Formato inequívoco:** El usuario debe introducir información en el sistema, pero puede que no esté familiarizado con el tipo de información a proporcionar o que desconozca el formato en el que la debe introducir.

- **Navegación entre escenarios:** El usuario es posible que requiera acceder a una gran cantidad de información que no se puede mostrar en un único escenario.
- **Similitud con el mundo real:** El usuario necesita saber cómo funciona un objeto de la interfaz que tiene una gran similitud con un objeto del mundo real.
- **Previsualización:** El usuario está buscando un elemento en una lista con varios componentes (al pulsar un botón examinar).
- **Favoritos:** El usuario necesita encontrar un mismo elemento de entre un conjunto con mucha frecuencia.
- **Área de acciones:** El usuario necesita conocer dónde encontrar las posibles acciones que se pueden realizar en el sistema y cómo activarlas.
- **Contenedor de navegaciones:** El usuario necesita encontrar un elemento de entre un conjunto de contenedores.
- **Ajuste de atributos:** El usuario quiere ver los atributos de los objetos con los que está trabajando y además quiere saber cómo modificarlos.
- **Aviso:** El usuario puede, de forma no intencionada, causar una situación problemática que se necesita resolver para continuar con la ejecución normal del sistema.
- **Indicaciones:** El usuario necesita saber en todo momento cómo seleccionar la funcionalidad del sistema.
- **Modo cursor:** A la hora de crear o modificar un objeto, el usuario debe saber qué función de edición se ha seleccionado previamente.
- **Navegador de una lista:** El usuario necesita navegar a través de una lista o modificar su contenido.

- **Filtrado continuo:** El usuario necesita encontrar un elemento en una lista ordenada.

2.3.2.2 Ventajas y limitaciones de la propuesta

La propuesta de Welie está orientada a métodos de desarrollo manual. La principal ventaja de esta propuesta es que dispone de una notación para representar la usabilidad. Esta notación está basada en el lenguaje de patrones. Otra de las ventajas consiste en que esta notación es aparentemente entendible por el usuario, ya que incluye ejemplos. Por tanto, el usuario puede formar parte del proceso de desarrollo porque es capaz de entender los patrones.

Aunque la orientación al usuario se consigue mediante la incorporación de ejemplos, la orientación hacia el analista tiene los mismos inconvenientes que presentan los patrones de Perzel, es decir, no describen cómo implementar los patrones en el sistema y las implicaciones que pueden tener los patrones con respecto a la arquitectura del sistema.

Otros de los inconvenientes de la propuesta de Welie son: no se especifica para qué tipo de tecnología se pueden utilizar los patrones que propone; no se especifican los atributos de usabilidad que se tratan en cada uno de los patrones.

2.3.3 Representación formal de patrones de usabilidad de Borchers

2.3.3.1 Definición

La mayoría de las propuestas de patrones de usabilidad son descripciones textuales y por lo tanto, no formales. Hay algunos autores, como Borchers, que proponen el uso de patrones definidos formalmente [Borchers, 2000]. Según Borchers, el uso de una notación formal para todos los patrones de usabilidad describiría los patrones con un lenguaje común para todos los autores que trabajaran en la definición de nuevos patrones. Borchers detecta que los diseñadores de interfaces tienen dificultades para explicar sus guías al resto de miembros del equipo de desarrollo. Una descripción formal de los patrones es menos ambigua para las partes implicadas en el

proceso de desarrollo a la hora de describir la estructura y el contenido de los patrones, por lo tanto, ayudaría a solucionar este problema.

La sintaxis que propone Borchers para la descripción de los lenguajes de patrones es un grafo acíclico dirigido donde cada uno de los nodos es un patrón. Además de la parte sintáctica de los patrones, Borchers define la semántica que se debe utilizar en la especificación de los patrones. La semántica está basada en los siguientes componentes:

- **Patrón:** Cada patrón representa un diseño de interfaz para solucionar un problema recurrente.
- **Contexto:** Cada patrón tiene un contexto representado por relaciones hacia patrones de mayor nivel de abstracción. Estas relaciones crean una jerarquía entre los patrones que forman el lenguaje de patrones.
- **Nombre:** El nombre del patrón ayuda a identificar rápidamente su funcionalidad.
- **Rango:** Indica cuánto de válido cree el analista que es el patrón.
- **Ejemplo:** Muestra un ejemplo típico del patrón para que el lector pueda entender su funcionalidad. Está pensado para que aquellos lectores que no sean expertos en el ámbito de la informática entiendan su funcionalidad.
- **Problema:** Describe cuál es el objetivo que se pretende alcanzar al utilizar el patrón.
- **Situación:** Muestra un ejemplo de las situaciones en las cuales el uso del patrón puede ser útil.
- **Solución:** Generaliza para cualquier sistema cual es la solución que aporta el uso del patrón.
- **Diagrama:** Representa de forma gráfica y resumida cuál es la solución del patrón. Para los expertos es más sencillo utilizar estos diagramas para comprender el uso de los patrones, ya que de un

simple vistazo tienen toda la información necesaria para entenderlos. El diagrama escogido dependerá del tipo de dominio: un boceto gráfico de la arquitectura de la interfaz, un pseudo código, diagramas UML [UML, 2008], bocetos de las historias de usuario, etc.

Además de proponer una notación formal para representar los patrones, y siguiendo la tendencia de Folmer y Bass, Borchers también afirma que los patrones de usabilidad se deben tener en cuenta desde las primeras etapas del proceso de desarrollo software.

2.3.3.2 Ventajas y limitaciones de la propuesta

La propuesta de Borchers está orientada a métodos de desarrollos manuales y formales. La principal ventaja de esta propuesta es que propone el uso de una notación específica, como es el uso de un lenguaje formal basado en patrones. Esta notación trata de ser una especie de estándar para que todos los autores que trabajen con patrones de usabilidad la utilicen. El propósito es bueno, pero es difícil poner de acuerdo a toda la comunidad científica para que adopte esta notación. A la hora de la verdad, cada autor usa su propia notación para describir los patrones que forman su propuesta. Además, la notación que propone Borchers es compleja ya que requiere que el analista que vaya a trabajar con los patrones de usabilidad tenga conocimientos básicos en métodos formales para ser capaz de entender la especificación de los patrones. Esta complejidad puede desembocar en que muchos autores de patrones de usabilidad no acepten dicha notación.

Otro inconveniente es que si ya es complicado entender una notación basada en formalismos para algunos analistas, aun más compleja es esta tarea para los usuarios finales. El alto grado de conocimiento que es necesario para entender los formalismos hace difícil que el usuario pueda entender las características de los patrones y seleccionar el que más le interese para sus aplicaciones.

Otros inconvenientes de la propuesta de Borchers son: no especifica para qué tecnología o plataforma podrían servir los patrones, no se incluyen pasos para guiar al analista; no se especifica cómo implementar la

propuesta en un lenguaje de programación; no se detallan los atributos de usabilidad que se abarcan con los patrones.

2.3.4 Colección de patrones de usabilidad de Brighton

2.3.4.1 Descripción

Otro de los autores que ha definido un pequeño conjunto de patrones de usabilidad es Griffiths, con lo que él ha definido como los patrones de usabilidad de Brighton [Griffiths, 2002]. El trabajo de Griffiths está más orientado al mundo industrial que al académico, ya que estos patrones carecen de publicaciones que destaquen en ambientes académicos. Aun así, son muy referenciados por el resto de autores dentro del contexto de los patrones de usabilidad, por lo que tienen un gran reconocimiento dentro de la comunidad científica. Los patrones que incluye Griffiths en su propuesta son:

- **Adelantarse a la escritura:** Los usuarios más experimentados ya conocen la información que se debe introducir en cada campo, el formato que debe seguir y la secuencia de órdenes necesarias. Este tipo de usuarios requieren que el ordenador procese sus órdenes de forma rápida y que el sistema refresque la información de la ventana de forma instantánea.
- **Salidas de emergencia:** Se debe evitar que el usuario tenga que visualizar un conjunto de diálogos para salir de las interfaces del sistema.
- **Avisar:** Se debe avisar a los usuarios en caso de que alguna de las acciones ordenadas por el usuario genere una situación de alerta, pero no debe acabar con la tarea que esté realizando en ese momento el usuario.
- **Retroalimentación de interacción:** Los usuarios deben conocer en todo momento que el sistema ha registrado un evento de interacción, tal como un clic de ratón, un movimiento de ratón, la pulsación de una tecla, etc.

- **Simplemente visible:** Hacer que el contenido de todo el formulario esté visible en la ventana, y hacer que la entrada de datos sobre la identificación personal se deje para el final.
- **Mostrar que el ordenador está pensando:** Mostrar retroalimentación al usuario para indicar que el sistema está funcionando, y si es posible, indicar el estado del proceso de ejecución que se ha alcanzado.
- **Mostrar el formato requerido:** Proporcionar al usuario detalles del formato requerido para los componentes de la ventana que requieren la entrada de información.
- **Arquitectura de modelo vista controlador:** Considerar la posibilidad de alternar modos y estilos de presentación e interacción que se puede requerir en las interfaces de usuario. Además, la interfaz se debe diseñar dentro de las posibilidades que ofrece una arquitectura basada en el modelo vista controlador [Krasner, 1998].
- **Piénsalo dos veces:** Para cada acción que el usuario pueda realizar, se debe considerar: la capacidad de que la acción sea reversible; la proporción de acciones reversibles que soporta el sistema; la frecuencia con que la acción se va a realizar; el grado de daño que puede causar una mala ejecución; la retroalimentación que debe proporcionar la acción (aviso, confirmación, autorización, etc.)
- **Tiempo para hacer algo más:** En caso de que el tiempo restante para finalizar una acción se pueda calcular, se debería mostrar al usuario. Esto se puede hacer mediante figuras o gráficos. Si el tiempo no se puede estimar pero el proceso tiene fases identificables, se debe indicar las fases que se han completado y las fases que restan.

En el trabajo, se muestra la relación existente entre los patrones mediante una matriz. En esta matriz se representa si la relación entre los patrones es directa o inversa.

2.3.4.2 Ventajas y limitaciones de la propuesta

La propuesta de Griffiths está orientada a métodos de desarrollo manuales. La principal ventaja es la existencia de una notación específica para representar la usabilidad. Esta notación está basada en el lenguaje de patrones. Además, se incluye una matriz para relacionar los distintos patrones entre sí.

Los patrones de Brighton son unos de los más completos porque abarcan un gran número de aspectos de usabilidad con implicaciones funcionales. A pesar de ser unos patrones orientados a ser utilizados en ambientes reales, los ejemplos que utilizan son poco descriptivos y no detallan las implicaciones que tienen estos patrones en la arquitectura del sistema. Se debería facilitar la labor del analista, detallándole cómo aplicar los patrones mediante pasos o un método.

Otro de los inconvenientes es que su descripción no está estructurada, sino que muestra toda la información de forma descriptiva sin dividirla en secciones o apartados. Esto dificulta la búsqueda de una característica concreta del patrón porque obliga a leerse toda la descripción para encontrarla. La información debería estar estructurada al modo tradicional de los patrones: en el título, una breve descripción, el problema que pretenden resolver y la solución que proponen. Además, esto dificulta la participación del usuario, el cual no puede formar parte del proceso de desarrollo a no ser que tenga amplios conocimientos de informática.

Otros de los inconvenientes de esta propuesta son: no se especifica para qué plataforma están pensados los patrones; no se especifica cómo implementar los patrones en un lenguaje de programación; no se especifican los atributos de usabilidad que abarca cada uno de los patrones.

2.3.5 Patrones de diseño de la interacción de Tidwell

2.3.5.1 Descripción

Otro de los autores que ha propuesto un conjunto de patrones similares a los de Welie y Perzel es Tidwell [Tidwell, 2005]. La nomenclatura que usa

Tidwell para los patrones es la de patrones de interacción, no de usabilidad, ya que estos patrones están más orientados a representar aspectos del diseño de la interfaz que a aspectos de usabilidad exclusivamente. Para Tidwell, los patrones de interacción son una ayuda para diseñar el modelo conceptual que hay detrás de la interfaz. Hay diseñadores (típicamente provenientes de la IPO) que empiezan el desarrollo del sistema por el diseño de la interfaz; primero diseñan la interfaz según las necesidades del usuario y después diseñan el modelo conceptual que hay detrás de esa interfaz. Por el contrario, hay otro grupo de diseñadores (típicamente provenientes de la Ingeniería del Software) que empiezan por el modelado conceptual y después eligen los patrones de interacción que representan lo expresado mediante los modelos conceptuales. De entre las dos técnicas, Tidwell se decanta por la primera, ya que el incorporar el diseño de la interfaz al diseño del sistema a posteriori implica normalmente cambiar parte del diseño del sistema.

A continuación se presentan los patrones que Tidwell propone para que los diseñadores representen la interfaz de forma abstracta y los problemas que resuelven. La notación utilizada para diseñar cada uno de estos patrones es muy gráfica, de forma que sea entendible por el usuario y pueda aportar mejoras a la usabilidad del sistema en fase de diseño:

- **Espacios navegables:** Un sistema tiene un gran número de contextos. Este patrón define cómo se organiza la navegación entre los distintos contextos.
- **Mapa de espacios navegables:** Este patrón proporciona un mecanismo para ayudar al usuario a conocer en todo momento en qué contexto está y a cuáles puede navegar desde ese contexto.
- **Ir un paso atrás:** Define la funcionalidad de regresar al contexto anterior a la última navegación (al paso anterior). Además, esta funcionalidad también incorpora la navegación de un contexto hacia delante (un paso adelante).
- **Historial de interacciones:** Este patrón representa la capacidad de almacenar la secuencia de acciones y navegaciones que realiza el usuario con el sistema y mostrárselo al usuario.

- **Marcadores:** Los sistemas de gran tamaño tienen muchos contextos. Los marcadores facilitan el acceso a los contextos más utilizados por el usuario de forma que éste pueda acceder rápidamente a ellos sin tener que realizar muchas navegaciones.
- **Mostrar acciones permitidas:** Este patrón indica cómo el sistema resalta aquellos elementos de un contexto que tengan asociada alguna acción que el usuario pueda realizar.
- **Breve descripción:** Este patrón define qué elementos de cada uno de los contextos van a llevar un texto descriptivo para clarificar la función que desempeña dicho elemento.
- **Inhabilitar elementos irrelevantes:** Este patrón se usa para determinar qué elementos de los contextos se deben mostrar inhabilitados porque su acción no está permitida.
- **Indicador de progreso:** Este patrón se usa en acciones que requieren un mecanismo para mostrar el tiempo restante que falta para finalizar su ejecución.

Una de las diferencias que existen entre los patrones de Tidwell y los de Perzel es en el número de patrones que existen en ambas propuestas. Esto se debe a que los patrones de Tidwell están descritos de forma más general y pueden englobar a varios de los patrones de Perzel. Por ejemplo, el patrón *Mapa de espacios navegables* de Tidwell engloba a los patrones: *Contexto del sitio*; *Usted se encuentra aquí* de Perzel.

2.3.5.2 Ventajas y limitaciones de la propuesta

La propuesta de Tidwell está orientada a métodos de desarrollo manual. La principal ventaja de esta propuesta es el uso de una notación específica para representar la usabilidad. Esta notación está basada en el lenguaje de patrones.

Como inconvenientes a la propuesta de Tidwell, cabe destacar que los patrones no incluyen ninguna descripción sobre cómo implementarlos y las implicaciones en la arquitectura del sistema que pueden tener. La descripción de los patrones sólo incluye el problema que resuelve y cuándo

usarlo, pero no da ninguna idea de cómo introducir su funcionamiento dentro de la arquitectura del sistema. Además, no se puede deducir qué patrones requieren menos esfuerzo por parte del analista a la hora de implementarlos.

Aunque la notación utilizada para representar cada uno de los patrones sea sencilla, no será entendible por aquellos usuarios que tengan un bajo nivel de conocimientos informáticos porque los ejemplos no son muy clarificadores. Por lo tanto, la participación del usuario en la fase de diseño está limitada a aquellos que ya han interactuado con otros sistemas software. No todos los usuarios van a poder construir sus propios modelos mentales en base a los artefactos visibles, y por lo tanto, no van a poder predecir el comportamiento de dichos artefactos.

Otros de los inconvenientes de los patrones de Tidwell son: no se especifica para qué tecnologías se podrían aplicar los patrones; no se incluyen pasos que guíen al analista en la aplicación de los patrones; no se especifican los atributos de usabilidad que abarca cada uno de los patrones.

2.3.6 Representación de los patrones de usabilidad mediante un repositorio de ontologías de Henninger

2.3.6.1 Definición

Tal y como se ha visto anteriormente, Borchers ha identificado unos cuantos problemas que se derivan del gran número de propuestas, pero no ha sido el único. Siguiendo esta tendencia, Henninger ha detectado una colección de problemas más detallada que la de Borchers [Henninger, 2005]:

1. Los patrones no están todos descritos con orientación hacia el problema de usabilidad que resuelven. No está claro en todos los casos qué problema de usabilidad resuelve cada uno de los patrones.
2. Después de seleccionar un patrón para incorporarlo al sistema, no es sencillo averiguar las relaciones entre otros patrones que se

podrían utilizar para complementar el patrón seleccionado. En algunos casos sí que hay relaciones entre los patrones de un mismo autor, pero son relaciones en base a navegaciones entre páginas Web. Además, no existen relaciones entre patrones desarrollados por distintos autores.

3. Los patrones están representados de manera informal en lenguaje natural, y en algunos casos esta definición es ambigua.
4. No está especificada la forma en la que los patrones pueden trabajar conjuntamente. La incorporación de un patrón al sistema puede afectar el funcionamiento de otros patrones.

El motivo por el que existen estos problemas es que todas las colecciones de patrones están basadas en repositorios pasivos. Los analistas deben conocer todos los patrones para aplicar el más adecuado a cada uno de los problemas de usabilidad, lo que requiere un gran esfuerzo por su parte analistas. Para resolver estos problemas, Henninger propone el uso de la Web Semántica para representar los distintos patrones de usabilidad [Henninger, 2005]. Estos autores han estudiado cómo se pueden usar los estándares de la Web Semántica para definir descripciones formales de patrones de usabilidad entendibles por los sistemas y por los analistas. La Web Semántica se basa en la idea de añadir metadatos semánticos a la Web. Esas informaciones adicionales (que describen el contenido, el significado y la relación de los datos) se deben proporcionar de manera formal, para que así sea posible evaluarlas automáticamente. Henninger et al usan una notación basada en ontologías para describir formalmente los patrones de usabilidad usando estándares de la Web Semántica. Una ontología es una descripción de los conceptos y relaciones entre los conceptos que están formalmente definidos dentro de un dominio de interés.

Los patrones de usabilidad definidos mediante ontologías se almacenan en un repositorio llamado BORE (Building an Organizational Repository of Experiences) [BORE, 2008]. BORE se utiliza como un framework para la búsqueda de patrones de usabilidad. Cuando el desarrollador selecciona un patrón para incorporarlo al sistema, esta herramienta busca las relaciones que el patrón seleccionado tiene con los otros patrones que forman el repositorio y devuelve una lista con los resultados de esta búsqueda para

que el analista sea consciente de las relaciones. La búsqueda se hace internamente a través de las ontologías en un proceso que es transparente para el analista. A partir de la lista de patrones relacionados, el analista puede detectar aquellos que son similares y utilizar de todos ellos el que más le interese dependiendo del sistema. Además, también permite detectar qué otros patrones se podrían incluir en el sistema para aumentar su usabilidad sin mayor esfuerzo.

La principal ventaja de usar una herramienta basada en la Web Semántica, como es BORE, es que la mayor parte del proceso de búsqueda de relaciones entre patrones se puede automatizar. El usuario debe responder a una serie de preguntas y en base a las respuestas, BORE determinará cuáles son los mejores patrones de usabilidad para satisfacer los requisitos del usuario. Otras de las ventajas de este proceso es que el usuario no necesita conocer en detalle cómo funcionan los patrones, sino simplemente debe responder a las preguntas que BORE le formule y esta herramienta se encargará del resto de las tareas. Previamente, el analista debe estudiar todos los patrones que conforman el repositorio para seleccionar el que más le convenga para cada caso. Una vez se haya elegido uno, BORE buscará las relaciones existentes con los otros patrones.

2.3.6.2 Ventajas y limitaciones de la propuesta

La propuesta de Henninger está orientada a métodos de desarrollo software manuales. La principal ventaja de esta propuesta es la existencia de una notación precisa para representar la usabilidad. Esta notación está basada en el lenguaje de ontologías para representar patrones de usabilidad. Otra de las ventajas es que propone el uso de una Web Semántica basada en ontologías como repositorio de patrones. Esta herramienta se puede considerar como una guía para facilitar la labor del analista.

Para que el analista pueda trabajar con esta Web Semántica, es necesario que los patrones de usabilidad existentes en el repositorio estén relacionados entre sí. El inconveniente aparece precisamente a la hora de crear el repositorio con el conjunto de patrones. La descripción de cada uno de los patrones depende del creador de dicho patrón, es decir, hay descripciones de patrones más detalladas que otras. Por lo tanto, la tarea de definir las relaciones entre los patrones es una tarea difícil (sobre todo

entre patrones de diferentes autores) y en algunos casos puede llegar a ser incluso subjetiva. La forma en la que el analista introduzca los patrones en el repositorio va a determinar el uso de esos patrones en el proceso de desarrollo.

Otros de los inconvenientes de la propuesta de Henninger son: no se especifica para qué plataforma se pueden utilizar los patrones; no se especifica cómo implementar los patrones en un lenguaje de programación; no se especifica para cada patrón los atributos de usabilidad que se abarcan; la representación de los patrones mediante ontologías hace que el usuario no pueda entender su funcionalidad y por tanto, no pueda participar en el proceso de desarrollo.

2.3.7 Patrones de interacción para el trabajo cooperativo

2.3.7.1 Descripción

Algunos autores han centrado su trabajo en definir patrones de interacción para construir sistemas cooperativos usables. Un sistema cooperativo es aquel en el que hay varios actores implicados y cada uno de estos actores interactúa de forma diferente. El estudio de las formas de interactuar en un sistema cooperativo se conoce como *Etnografía*. En este contexto se encuentran los patrones definidos en el proyecto Pointer [Pointer, 2008]. El proyecto trata de estudiar la idoneidad de patrones para representar la información que fluye entre las distintas personas que deben interactuar con un mismo sistema.

Históricamente, los estudios etnográficos se han desarrollado en una gran variedad de sistemas cooperativos, como por ejemplo, sistemas de tráfico aéreo, reserva de habitaciones, urgencias de un hospital, bancos, etc. El objetivo de estos estudios era conocer en qué forma se utilizaba la tecnología como herramienta para la cooperación. En base a estos estudios, el proyecto Pointer recoge una lista de patrones de interacción que deberían estar presentes en cualquier sistema cooperativo y usable. Los patrones son:

- **Artefacto como pista de auditoria:** Este patrón representa la forma en la que se puede utilizar al sistema como un mecanismo para almacenar las acciones que realizan los usuarios. Es decir, el sistema actúa como coordinador entre los usuarios, permitiéndoles localizar quién ha hecho cada tarea.
- **Representación múltiple de la información:** Mediante este patrón se representan las distintas formas de las que disponen los usuarios para visualizar la información del sistema. Cada una de las formas de visualización puede ser más conveniente para cada una de las tareas que realizan los usuarios en el sistema.
- **Artefacto público:** Este patrón se utiliza para controlar las tareas que cada uno de los usuarios del sistema están haciendo en el instante actual. Esto muestra una perspectiva global de todas las acciones que se están desarrollando en el sistema.
- **Representar un artefacto invisible:** Este patrón representa la forma en la que uno de los usuarios puede habilitar o inhabilitar los artefactos con los que interactúan el resto de usuarios del sistema. Estas tareas se pueden considerar como parte de las tareas de administración del sistema.
- **Trabajar con interrupciones:** Este patrón se centra en representar las posibles interrupciones que se pueden dar en un lugar de trabajo. Trata con situaciones donde las interrupciones aparecen de varias fuentes y están fuera del control de los usuarios.
- **Colaboración en pequeños grupos:** Este patrón representa en qué forma interactúan grupos de pequeños usuarios, es decir, se especifica qué artefactos del sistema estarán disponibles para realizar la colaboración entre usuarios de un mismo grupo.
- **Recepción de información:** Este patrón se utiliza para centralizar y filtrar la información que le llega al sistema. Actúa del mismo modo que un recepcionista de un edificio. Cada vez que llega

información al sistema hace de coordinador y la envía al usuario que la necesita.

- **Dar un paseo:** Este patrón se utiliza para representar la capacidad de los usuarios de utilizar áreas del sistema para ver las actividades de otros grupos de usuarios, como históricos, qué acciones se están ejecutando en la actualidad, el número de usuarios conectados, etc.
- **Coincidencia en responsabilidades:** Este patrón se utiliza para representar las superposiciones existentes entre los roles de los distintos grupos de usuarios del sistema. Es decir, hay roles que debe desempeñar cada uno de los grupos que pueden coincidir en dos o más grupos.
- **Asistencia a través de la experiencia:** Dentro de los grupos de usuarios, habrá un subgrupo con más experiencia en el desempeño de sus roles que otros. Este subgrupo de gente experta debería poder ayudar a la gente que se está iniciando en sus tareas y por lo tanto tiene menos experiencia. Con este patrón se representa en qué forma los usuarios más avanzados ayudan a los que aun no han adquirido experiencia.

2.3.7.2 Ventajas y limitaciones de la propuesta

La propuesta del proyecto Pointer está orientada a métodos de desarrollo software manuales. La principal ventaja de esta propuesta es la existencia de una notación para representar la usabilidad. Esta notación está basada en el lenguaje de patrones. Otra de las ventajas es que el ámbito de aplicación de los patrones está perfectamente especificado: sistemas cooperativos.

Cada uno de los patrones va acompañado de una descripción textual, en el que se incide porqué es útil su uso, en qué situaciones se debe utilizar, y las implicaciones de diseño que tiene su incorporación al sistema. De toda esta información, la que más carece de precisión es la que incide en las implicaciones de diseño que tiene el uso del patrón. No aparece ninguna descripción rigurosa que indique de forma genérica los cambios en la arquitectura del sistema que acarrea la incorporación del patrón ni tampoco

información sobre cómo implementarlo. Además, tampoco se puede saber qué patrones requieren más esfuerzo por parte del analista para su implementación.

Uno de los aspectos que se podría mejorar es la facilidad del lector para entender los patrones. La descripción textual es adecuada, pero se podrían utilizar ejemplos de sistemas reales para ver de forma práctica su uso. Estos ejemplos también facilitarían la incorporación del usuario en el proceso de desarrollo.

Por último, cabe destacar como inconvenientes detectados los siguientes: no existen pasos para guiar la labor del analista; aunque muestra las relaciones que hay entre los distintos patrones, no se especifican los atributos de usabilidad que abarca cada uno de los patrones.

2.3.8 Conclusiones

Esta sección ha mostrado los trabajos dedicados a describir patrones de usabilidad. Son muchas las propuestas existentes en el ámbito de los patrones de usabilidad, al igual que las propuestas sobre la incorporación de la usabilidad a nivel de requisitos. En cambio, las propuestas de patrones de usabilidad son mucho más homogéneas entre sí que las propuestas relacionadas con la incorporación a nivel de requisitos.

Los patrones estudiados consisten básicamente en una serie de descripciones textuales de mecanismos que ayudan a mejorar la usabilidad de los sistemas. Tal y como están descritos los patrones, éstos se pueden incorporar en cualquiera de las fases del proceso de desarrollo software, aunque la mayoría de autores coinciden en la importancia de incorporarlos en la fase más temprana posible para evitar tener que modificar la arquitectura del sistema.

Todas las propuestas tienen en común que están descritas de manera textual. Esto implica que la búsqueda de características concretas de los patrones requiera la lectura de toda la documentación. Algunos patrones, como los de Welie, están bastante estructurados y las búsquedas se pueden hacer de forma rápida, mientras que otros como los de Brighton carecen de un esquema claro y las búsquedas son más costosas.

Por otro lado, los patrones estudiados están descritos de manera muy cercana a la implementación del sistema. Además, muchos de estos patrones (como los de Welie y Perzel) incluyen en su descripción posibles capturas de pantalla. En cambio, ninguno de los trabajos da detalles sobre cómo implementar la funcionalidad del patrón. Se podría dar una descripción en pseudocódigo o mediante algún algoritmo que facilitara la labor del implementador.

Para concluir, los patrones aquí descritos son aplicables a cualquier proceso de desarrollo software, incluido MDD. En cambio, esta incorporación no es directa. Para que la incorporación en un proceso MDD se pueda realizar es necesario aumentar el nivel de abstracción de estos patrones a nivel de modelado conceptual. Éste es el trabajo de investigación de la tesis: detectar primitivas conceptuales a partir de las cuales representar aspectos de usabilidad. Por lo tanto, los patrones de usabilidad de la literatura son una buena base a la hora de estudiar las características de usabilidad a tener en cuenta en los modelos conceptuales.

2.4 Usabilidad en entornos MDD

El último conjunto de autores referenciado en esta tesis es el formado por aquellos que han tratado de incorporar la usabilidad en las primeras etapas del proceso de desarrollo software en entornos MDD [Mellor, 2002]. En principio, el objetivo de estos trabajos es justamente el mismo que el que se pretende alcanzar con esta tesis: incorporar la usabilidad en el modelo conceptual de un método MDD. Así como en las secciones donde se han explicado los trabajos existentes para incorporar la usabilidad en la arquitectura, a nivel de requisitos y mediante patrones hay una gran variedad de trabajos relacionados, a la hora de buscar trabajos que propongan la incorporación de la usabilidad en un entorno MDD los trabajos existentes hoy en día son muy escasos. Esto es debido a que es un tema reciente, ya que hasta hace poco la preocupación principal de la investigación en entornos MDD estaba centrada en conseguir aplicaciones funcionales a partir de los modelos conceptuales, y no aplicaciones usables. Así, es importante recalcar la originalidad de la presente tesis, que intenta aportar una solución a un problema que actualmente está poco tratado. A

continuación se presentan los trabajos más relevantes que existen hoy en día para incorporar la usabilidad en el ámbito de los modelos conceptuales de las tecnologías MDD.

2.4.1 Modelado de la usabilidad con un Diagrama de Transición entre Estados

2.4.1.1 Definición

En ambientes de diseño de software, el Diagrama de Transición entre Estados (DTE) se utiliza para describir el comportamiento de los sistemas. En cambio, algunos autores han utilizado también este diagrama para describir la usabilidad del sistema. En este contexto se encuentra el trabajo de Tao [Tao, 2005]. Según este autor, los DTEs se pueden utilizar para capturar las interacciones y el contexto en el cual tiene lugar la interacción. Esta aproximación permite separar el diseño de la interacción de su parte visual, por lo que está alineada con la idea de la IS de separar el modelado de la funcionalidad del de la interfaz.

Además de la notación básica de los DTEs que incluye el concepto de estados, transiciones entre estados, eventos y acciones, existen otros elementos que permiten la composición de estados, es decir, estados que contienen subestados. Según Tao, una composición de estados puede representar una pantalla, mientras que un subestado puede representar un modo de interacción. Los eventos deben ser observables como resultado de la interacción entre el usuario y el sistema, y una secuencia de eventos describe los pasos que el usuario lleva a cabo para completar una tarea.

Tao afirma que no es el único autor que ha propuesto modelar la usabilidad mediante DTEs, sino que autores de la importancia de Nielsen también han detectado la necesidad de modelar el flujo de información. Algunos de los diez heurísticos de usabilidad de Nielsen [Nielsen 1993] no están basados en la parte visible de la interfaz de usuario sino en el flujo de información que se desprende en la interacción con el usuario. Es decir, estos heurísticos se pueden aplicar al modelo de comportamiento aun cuando la parte visible no esté disponible.

El uso de los DTEs para modelar la usabilidad se ha utilizado empíricamente con estudiantes. Esta notación permitió a los estudiantes aplicar los principios de usabilidad para la detección de errores y mejorar el diseño de la interfaz de usuario. También ayuda a entender la naturaleza de algunos de los errores comunes entre los estudiantes.

2.4.1.2 Ventajas y limitaciones de la propuesta

La propuesta de Tao está orientada a métodos de desarrollo manuales. La principal ventaja de esta propuesta es la existencia de una notación específica para representar la usabilidad. Esta notación está basada en el uso de DTEs.

La propuesta de Tao está centrada en capturar los flujos de interacción que se llevan a cabo en el sistema. Esta aproximación es adecuada para sistemas en los que haya flujos de interacción muy complejos, pero carece de sentido en aquellos sistemas en los que las interacciones tengan pocos pasos o sean acciones sencillas. Por tanto, se puede afirmar que la propuesta de Tao no puede por sí sola representar toda la usabilidad del sistema. Se puede utilizar como complemento para aquellos sistemas con flujos de interacción complejos, pero es necesario otros modelos para representar de forma abstracta la usabilidad sin que dependa de flujos de información.

Otro de los inconvenientes de esta propuesta es que para entender los DTEs, es necesario tener conocimientos en el área de la informática y por lo tanto, la usabilidad representada mediante DTEs no podrá ser evaluada por los usuarios hasta que no se implemente el sistema.

Por último, también cabe destacar los siguientes inconvenientes: no se definen guías o pasos para la construcción de los DTEs, no se especifica cómo implementar los DTEs en un lenguaje de programación; no se especifica los atributos de usabilidad que se podrían representar mediante los DTEs.

2.4.2 Evaluación temprana de la usabilidad en MDA

2.4.2.1 Descripción

Entre los autores que han propuesto la incorporación de la usabilidad en entornos MDD, existen algunos que se han centrado en propuestas para el estándar MDD, es decir, en MDA [MDA, 2008]. Uno de los trabajos más relevantes en este ámbito es el de Abrahao et al [Abrahao, 2006] [Abrahao, 2005] [España, 2006]. En su trabajo, Abrahao propone un Modelo de Usabilidad a partir del cual se puede evaluar y mejorar la usabilidad representada en el nivel *Platform Independent Model (PIM)* de MDA. Como resultado de esta evaluación se genera una lista con los problemas de usabilidad encontrados. Mediante esta lista, el analista puede hacer los cambios oportunos en el nivel PIM para corregir las carencias detectadas en usabilidad o a nivel de *Platform Specific Model (PSM)* modificando las reglas de transformación que generan el código que implementa el sistema desde el modelo conceptual.

Este Modelo de Usabilidad se ha centrado en la definición de usabilidad proporcionada por la ISO/IEC 9126-1 [ISO/IEC 9126-1, 2001]. Esta decisión estuvo motivada porque, según los autores, este estándar es más apropiado para una evaluación temprana de la usabilidad ya que trata con características propias del producto y se puede utilizar para evaluar artefactos PIM. Además de la ISO/IEC 9126-1, Abrahao ha utilizado características de la usabilidad definidas por otros autores de la comunidad IPO, como los criterios ergonómicos de Bastien y Scapin [Bastien, 1993].

El Modelo de Usabilidad establece relaciones entre atributos de usabilidad y elementos PIM que se ven afectados por ellos. El propósito es describir qué elementos del modelo contribuyen a la satisfacción de ciertos atributos de usabilidad. El Modelo de Usabilidad, tal como propone el estándar ISO/IEC 9126-1, parte de la característica de calidad *Usabilidad* y la descompone en varias subcaracterísticas. A su vez, las subcaracterísticas se descomponen en atributos sobre los que se puede aplicar métricas para obtener su valor de usabilidad. El Modelo de Usabilidad está compuesto por las siguientes subcaracterísticas y atributos:

- **Facilidad de aprendizaje:** Esta subcaracterística representa la capacidad del producto software que permite al usuario aprender el funcionamiento del sistema. Tiene como atributos: *Facilidades de ayuda* (asistentes, documentación, etc.); *Predecibilidad*, que se refiere a la facilidad en que el usuario puede determinar el resultado de sus acciones futuras; *Realimentación informativa*, en respuesta a las acciones del usuario; *Facilidad de memorización*, que representa cuánto de rápido y preciso pueden los usuarios recordar cómo usar un sistema después de haberlo utilizado.
- **Comprensibilidad:** Es la capacidad del producto software de permitir al usuario comprender si el producto software es adecuado y cómo puede ser utilizado para tareas particulares. Esta subcaracterística tiene los siguientes atributos: *Legibilidad* de texto e imágenes; *Facilidad de lectura*, que a su vez incluye *Agrupación cohesiva de la información* y *Densidad*; *Familiaridad*, que es el grado en que el usuario reconoce los componentes de la interfaz y ve la interacción como algo natural; *Brevidad*, que está relacionado con la reducción del esfuerzo cognitivo; Finalmente, el atributo *Orientación al usuario* está compuesto por los atributos *Calidad de los mensajes* y *Navegabilidad*.
- **Operabilidad:** Es la capacidad del producto software que permite al usuario operar con él y controlarlo. Esta subcaracterística está formada por los siguientes atributos: *Capacidad de instalación*, que representa la ayuda que proporciona el sistema para su instalación; *Validación de datos*; *Capacidad de control*, que representa el grado de control que el usuario tiene sobre los servicios; *Capacidad de adaptación*, que está formado por *Adaptabilidad* (la capacidad del sistema para que el usuario lo adapte a sus necesidades) y *Adaptatividad* (la capacidad del sistema para adaptarse por sí solo a las necesidades del usuario).
- **Grado de atracción:** Esta subcaracterística representa la capacidad del producto software de ser atractivo para el usuario. Está compuesta por atributos que miden la uniformidad del color de las interfaces, del estilo, la fuente y la posición de los elementos en la pantalla.

- **Conformidad:** Esta subcaracterística es la capacidad del producto software de seguir estándares, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad. Los atributos son: *Cumplimiento de ISO/IEC 9126-1; Cumplimiento con ISO/IEC 9241-10 [ISO/IEC 9241-10, 1998]; Cumplimiento con guía de estilo de Java; Cumplimiento con guía de estilo de Microsoft.*

<ul style="list-style-type: none"> 1.Learnability 1.1.Help Facilities 1.1.1.Documentation Completeness 1.1.2.Multi-user Documentation <ul style="list-style-type: none"> 1.1.2.1.Capability Profiled 1.1.2.2.Role Profiled 1.2.Predictability 1.2.1.Icon Significance 1.2.2.Icon/Link Title Significance 1.2.3.Action Determination 1.3.Informative Feedback 1.4.Memorability 1.4.1.Time to Remember 1.4.2.Accuracy 2.Understandability 2.1.Legibility 2.1.1.Font Size 2.1.2.Contrasting Text 2.1.3.Disposition 2.2.Readability 2.2.1.Information Grouping Cohesiveness 2.2.1.1.Location Grouping 2.2.1.2.Format Grouping 2.2.2.Information Density 2.3.Familiarity 2.3.1.Labeling Significance 2.3.2.Internationalization 2.3.3.Metaphor 2.4.Workload Reduction 	<ul style="list-style-type: none"> 3.Operability 3.1.Installability 3.1.1.Ease of Installation 3.1.2.Multiplicity of Installation 3.1.3.Updateability 3.1.4.Update Transparency 3.2.Data Validity 3.3.Controlability 3.3.1.Edition Deferral 3.3.2.Cancel Support 3.3.3.Explicit Execution 3.3.4.Interruption Support 3.3.5.Undo Support 3.3.6.Redo Support 3.4.Capability of Adaptation 3.4.1.Adaptability 3.4.2.Adaptivity 3.5.Consistency 3.5.1.Steady Behavior of Controls 3.5.2.Permanence of Controls 3.5.3.Stability of Controls 3.5.4.Order Consistency 3.5.5.Label Consistency 3.6.Error Management 3.6.1.Error Prevention 3.6.2.Error Recovery 3.7.State System Monitoring 4.Attractiveness 4.1.Background Color Uniformity 4.2.Font Color Uniformity
---	--

2.4.1.Brevity	4.3.Font Style Uniformity
2.4.1.1.Values Inizialization	4.4.Font Size Uniformity
2.4.1.1.1.Initial Values Completion	4.5.UI Position Uniformity
2.4.1.1.2.Initial Values Modifiability	4.6.Subjective Appealing
2.4.1.2.Actions Minimization	5.Compliance
2.4.2.Self-descriptiveness	5.1.Degree of Fulfillment with the ISO/IEC 9126
2.5.User Guidance	5.2.Degree of Fulfillment with the ISO 9241-10
2.5.1.Message Quality	5.3.Degree of Fulfillment with the Microsoft style guide
2.5.2.Navigability	5.4.Degree of Fulfillment with the Java style guide

Tabla 2.1 Modelo de Usabilidad de Abrahao [Abrahao, 2006] [Abrahao, 2005] [España, 2006].

La Tabla 2.1 muestra en detalle todas las subcaracterísticas y los atributos del Modelo de Usabilidad propuesto por Abrahao. En base a este Modelo de Usabilidad, Abrahao propone algunas métricas para los atributos de dicho modelo de forma que se pueda calcular el valor de usabilidad del sistema a nivel PIM dentro del proceso MDA. De esta forma se consigue una evaluación temprana y automática, lo que produce una reducción de costes en la evaluación de la usabilidad. Además, esta aproximación permite que el analista pueda reparar los defectos detectados en la usabilidad simplemente modificando el nivel PIM. En el trabajo, los autores aplican este modelo a un caso de ilustración en el que evalúan la usabilidad del sistema.

2.4.2.2 Ventajas y limitaciones de la propuesta

La propuesta de Abrahao está centrada en métodos de desarrollo software basados en las transformaciones entre modelos. Como ventaja de esta propuesta, destaca la existencia de una notación para representar la usabilidad. Esta notación está basada en un Modelo de Usabilidad que agrupa la mayoría de los atributos de usabilidad existentes. Otra de las ventajas es que prácticamente tiene en cuenta todos los atributos de usabilidad definidos en la literatura.

En cambio, el Modelo de Usabilidad planteado por Abrahao tiene también una serie de inconvenientes. Muchos de los atributos del Modelo de Usabilidad no parecen poder medirse a nivel de PIM, ya que son totalmente subjetivos y dependientes del usuario. En este conjunto están por ejemplo los atributos *Familiaridad de conceptos* y *Calidad de los mensajes*. Una de las tareas que restan por realizar sería precisamente la de clasificar los atributos del modelo que son medibles a nivel de PIM y los que no.

Una vez llevada a cabo esta clasificación, el siguiente paso sería el de definir las métricas para aquellos atributos medibles a nivel de PIM. Aunque los autores han propuesto algunas métricas, el porcentaje de métricas definidas es escaso. Además, las métricas no están explícitamente relacionadas con el modelo conceptual del método de desarrollo MDA utilizado en su trabajo. Por lo tanto, habría que establecer relaciones entre las métricas propuestas y elementos del nivel PIM.

El Modelo de Usabilidad habría que validarlo empíricamente. Abrahao ha realizado casos de estudio usando el Modelo de Usabilidad, pero la evaluación no se ha llevado a cabo sobre el modelo conceptual (nivel PIM) sino sobre la aplicación final ya generada, y por lo tanto no se puede saber si las métricas proporcionan un valor de usabilidad cercano al percibido por el usuario. Habría que validar el Modelo de Usabilidad realizando una evaluación sobre el nivel PIM y posteriormente contrastarlo con el valor de usabilidad percibido por el usuario. De esta forma se sabría si la evaluación de la usabilidad temprana coincide con la proporcionada por los tests de usuario.

Otra de las carencias del trabajo es que no se muestran las relaciones que pueden existir entre los diferentes atributos. Es decir, al mejorar la usabilidad de un atributo se puede mejorar o empeorar la usabilidad de otros atributos que están relacionados con el primero. Es conveniente tener un mapa con todas las relaciones posibles para que el analista sea consciente de las repercusiones que tiene mejorar la usabilidad de cada uno de los atributos.

Algunas propuestas para mejorar estos inconvenientes se pueden encontrar en [Panach, 2007, a] y [Panach, 2008].

Por último, también se han detectado los siguientes inconvenientes: no se especifica para qué tipo de plataforma se podría aplicar el Modelo de Usabilidad; no se incluyen pasos o un método para guiar la labor del analista; no se especifica cómo implementar los atributos del Modelo de Usabilidad en un lenguaje de programación; no existen ejemplos entendibles por el usuario para que éste pueda ser capaz de decidir los atributos de usabilidad que desea en su sistema.

2.4.3 Conclusiones

La incorporación de la usabilidad en los métodos MDD es justamente el principal objetivo de esta tesis. De ahí que los trabajos descritos en esta sección se comparen de forma especial con nuestra investigación. Tal y como se puede apreciar, el número de trabajos existentes sobre la incorporación de la usabilidad en un proceso de desarrollo MDD es muy escaso. Por lo tanto, se puede concluir que el objetivo de esta tesis está ubicado en un marco novedoso y poco trabajado, pero no exento de importancia para la generación de sistemas de calidad mediante métodos MDD.

La diferencia principal entre la propuesta de esta tesis y las existentes en la literatura reside en la profundidad del trabajo. Los autores de la literatura han planteado la idea de modelar la usabilidad a nivel conceptual y los beneficios que ello conlleva, pero no han detallado cómo llevarlo a cabo. Sus propuestas son demasiado abstractas y no están aplicadas a un método MDD específico en el cual se pueda validar empíricamente la mejora en la usabilidad con respecto a otros métodos MDD que no soporten el modelado de la usabilidad. Esta tesis aborda la comparativa mediante una prueba experimental en la que se compara, por un lado, un sistema generado con un método MDD al que se han añadido Primitivas Conceptuales que representan la usabilidad, y por otro lado, el mismo sistema desarrollado con otro método MDD que no soporta el modelado de la usabilidad. Mediante esta prueba se puede evaluar la mejora en usabilidad conseguida con la incorporación de características de usabilidad a nivel conceptual.

Además, ninguno de los autores de la literatura ha definido primitivas conceptuales con las que representar la usabilidad de forma abstracta. En

sus propuestas indican cómo se podría modelar la usabilidad pero no llegan a definir primitivas conceptuales ni a aplicar el modelado a un caso de ilustración o a un experimento de laboratorio. Por todo ello, consideramos que esta tesis aporta soluciones a varias limitaciones de los trabajos relacionados del mismo ámbito.

2.5 Visión general del estado de la cuestión

Una vez estudiados en profundidad los trabajos que tratan la incorporación de la usabilidad en el sistema, esta sección presenta una comparativa entre todas esas propuestas. La Tabla 2.2 presenta esta comparación en base a los criterios de valoración comentados al principio del capítulo (página 27).

Del estudio realizado, cabe destacar que la mayoría de propuestas están orientadas a métodos de desarrollo tradicionales, es decir, no está pensada para métodos MDD. Además, pocas son las propuestas que definen una notación precisa para representar la usabilidad de forma abstracta y que definen un método para incorporar esa usabilidad dentro del proceso de desarrollo. Estas carencias son precisamente las que se quieren resolver en la tesis. Para ello se presenta un método para incorporar la usabilidad en un entorno de desarrollo MDD. Al final de la Tabla 2.2 se ha añadido una entrada para estudiar la propuesta de la tesis con los criterios de evaluación utilizados en los trabajos presentados en el estado de la cuestión.

El método propuesto en la tesis está pensado para aplicaciones de gestión, tanto para escritorio como para aplicaciones Web. Además, el método consta de una serie de pasos bien definidos que se pueden aplicar a cualquier método de desarrollo MDD. La usabilidad se representa en base a primitivas conceptuales y las guías para implementar la usabilidad se especifican con Diagramas de Clases y de Secuencia. Para cada primitiva conceptual se va a especificar los atributos de usabilidad con los que está relacionada. Por último, la participación del usuario en el proceso de desarrollo es alta. Aunque las primitivas conceptuales no son entendibles por el usuario, la evaluación se puede realizar con la aplicación final, ya

que la generación a partir de los modelos conceptuales es automática y por tanto se puede generar el sistema rápidamente. En caso de que alguna característica de usabilidad no satisfaga los requisitos del usuario, se podrán hacer los cambios oportunos en el modelo conceptual, volver a generar, y el usuario podrá volver a hacer una nueva evaluación en poco tiempo. En los siguientes capítulos de la tesis se abordan todas estas cuestiones.

Autor	Tecnología y ambientes	Incluye pasos o guías en la propuesta	Uso de modelos conceptuales	Notación precisa para representar la usabilidad	Especifica cómo implementar la propuesta	Cantidad de atributos de usabilidad tratados	Grado de comprensibilidad por el usuario
Comstock	No especifica	Sí, da una lista de buenas prácticas	Está pensada para métodos que usen modelos conceptuales	No	No	No especifica los atributos relacionados con las buenas prácticas	Poca
Bass	No especifica	Incluye matriz que relaciona beneficios de usabilidad con la arquitectura	Está más orientada a métodos tradicionales	No	No	Pocos	Poca
Folmer	No especifica	No	Está más orientada a métodos tradicionales	No	No	No especifica los atributos	Poca

Autor	Tecnología y ambientes	Incluye pasos o guías en la propuesta	Uso de modelos conceptuales	Notación precisa para representar la usabilidad	Especifica cómo implementar la propuesta	Cantidad de atributos de usabilidad tratados	Grado de comprensibilidad por el usuario
Lauesen	No especifica	Sí, indica qué características de usabilidad están en cada estilo	Está más orientada a métodos tradicionales	No	No	Sólo usa 9 atributos	Media
Bevan	No especifica	Define pasos pero no especifica cómo llevarlos a cabo	Está más orientada a métodos tradicionales	No	No	No especifica los atributos	Alta (usa prototipos del sistema para la evaluación con usuarios)
Cysneiros	No especifica	No, sólo propone el uso de i* pero no guía su construcción	Está más orientada a métodos tradicionales	Notación i*	No	No especifica los atributos	Poca

Autor	Tecnología y ambientes	Incluye pasos o guías en la propuesta	Uso de modelos conceptuales	Notación precisa para representar la usabilidad	Especifica cómo implementar la propuesta	Cantidad de atributos de usabilidad tratados	Grado de comprensibilidad por el usuario
Adikari	No especifica	No, sólo propone usar Modelo de Usuario y de Diseño, sin especificar cómo	Está más orientada a métodos tradicionales	No	No	No especifica los atributos	Poca
Juristo	No especifica	Sí, da guías para la captura de requisitos de usabilidad	Está más orientada a métodos tradicionales	Propone el uso de Diagramas de Clase, de Secuencia y patrones	Da una breve descripción	Atributos relacionados con aspectos funcionales	Alta (las guías contienen preguntas entendibles)
Perzel	Ambientes Web	No	Está más orientada a métodos tradicionales	Lenguaje basado en patrones	No	No especifica los atributos	Baja (no usa ejemplos)

Autor	Tecnología y ambientes	Incluye pasos o guías en la propuesta	Uso de modelos conceptuales	Notación precisa para representar la usabilidad	Especifica cómo implementar la propuesta	Cantidad de atributos de usabilidad tratados	Grado de comprensibilidad por el usuario
Welie	No especifica	No	Está más orientada a métodos tradicionales	Lenguaje basado en patrones	No	No especifica los atributos	Media (los ejemplos son entendibles, pero la descripción no)
Borchers	No especifica	No	Está más orientada a métodos tradicionales y formales	Lenguaje formal basado en patrones	No	No especifica los atributos	Muy poca
Brighton	No especifica	No, sólo incluye una matriz para relacionar los patrones	Está más orientada a métodos tradicionales	Lenguaje basado en patrones	No	No especifica atributos	Baja

Autor	Tecnología y ambientes	Incluye pasos o guías en la propuesta	Uso de modelos conceptuales	Notación precisa para representar la usabilidad	Especifica cómo implementar la propuesta	Cantidad de atributos de usabilidad tratados	Grado de comprensibilidad por el usuario
Tidwell	No especifica	No	Está más orientada a métodos tradicionales	Lenguaje basado en patrones	No	No especifica atributos	Baja (usa ejemplos pero son demasiado abstractos)
Henninger	No especifica	Define una herramienta que incluye guías para el analista	Está más orientada a métodos tradicionales	Lenguaje de patrones usando ontologías	No	No especifica atributos	Baja
Pointer	Sistemas cooperativos	No	Está más orientada a métodos tradicionales	Lenguaje basado en patrones	No	No especifica atributos	Baja

Autor	Tecnología y ambientes	Incluye pasos o guías en la propuesta	Uso de modelos conceptuales	Notación precisa para representar la usabilidad	Especifica cómo implementar la propuesta	Cantidad de atributos de usabilidad tratados	Grado de comprensibilidad por el usuario
Tao	No especifica	No, sólo propone el uso de DTEs pero no guía su construcción	Está más orientada a métodos tradicionales	Notación DTE	No	No especifica atributos	Baja
Abrahao	No especifica	No	MDD	Modelo de Usabilidad	No	Aborda la mayoría de atributos existentes	Baja
Panach	Web y escritorio	Sí, método MIMAT	MDD	En base a Primitivas Conceptuales	Sí, en base a Diagramas de Clase y de Secuencia	Especifica los atributos de usabilidad tratados	Alta (el usuario puede interactuar con el sistema generado)

Tabla 2.2 Comparación entre todos los autores estudiados en el estado de la cuestión

Capítulo 3

Planteamiento del Problema

Este capítulo presenta el problema que ha motivado la tesis. Con el objetivo de resolver dicho problema, se explica con especial detalle aquellos conceptos ya existentes en la literatura en los que se ha basado la solución planteada en la tesis. Por último, se comenta lo que se entiende por método de desarrollo MDD y en especial, por las TTMs.

3.1 Problema a resolver

Hoy en día existen propuestas para incorporar la usabilidad desde las primeras etapas del proceso de desarrollo software, tal y como proponen Bass [Bass, 2003] y Folmer [Folmer, 2004] en sus trabajos. Esta tesis se centra en la propuesta de Juristo [Juristo, 2007, a], quien propone incorporar la usabilidad desde la etapa de captura de requisitos. Para ello, Juristo ha definido una serie de características de usabilidad llamadas **Functional Usability Features (FUF)**. Estas características no sólo afectan a la interfaz del sistema, sino que tienen implicaciones en el diseño de su arquitectura. Los FUFs se dividen en varios subtipos llamados **mecanismos de usabilidad**. De ahí que sea importante detectar los requisitos de estos mecanismos antes de diseñar la arquitectura con el fin de evitar modificaciones.

Para capturar los requisitos de usabilidad con implicaciones funcionales, Juristo ha definido unas guías con preguntas que el analista debe preguntar al usuario. Estas guías están pensadas para indicar al analista las

características de usabilidad que debe incorporar a la arquitectura del sistema. Además, Juristo describe una aproximación basada en Diagramas de Clases y de Secuencia para incluir los mecanismos de usabilidad en el diseño del sistema.

El principal problema de la propuesta de Juristo es que todo el proceso desde la captura de requisitos hasta la implementación se tiene que hacer manualmente. Esto puede ralentizar el proceso de desarrollo hasta el punto de que el analista tenga que sopesar el beneficio de incluir los mecanismos de usabilidad con el coste y decidir si conviene tratar la usabilidad.

Para resolver este problema, la tesis plantea incorporar los mecanismos de usabilidad dentro de un entorno de desarrollo MDD. El analista sólo tendría que modelar las características de usabilidad en un modelo conceptual, y el diseño de la arquitectura y la implementación se automatizarían gracias al soporte proporcionado por un compilador de modelos. De esta manera, el coste de incorporar mecanismos de usabilidad sería mínimo, pudiendo generar aplicaciones usables eficientemente.

3.2 Contribuciones con respecto al estado de la cuestión

Tal y como se ha comentado en el capítulo 2 (Estado de la Cuestión), son muchos los trabajos que tratan de incorporar la usabilidad desde las primeras etapas del proceso de desarrollo software. Estos trabajos se pueden clasificar en cuatro grupos. Un primer grupo está formado por los trabajos que incluyen la usabilidad a nivel de arquitectura. Estos trabajos presentan como principal inconveniente que no detallan los cambios que habría que aplicar en la arquitectura para incorporar características de usabilidad. Un segundo grupo propone incorporar la usabilidad a nivel de captura de requisitos, pero son propuestas que añaden mucha complejidad al proceso de desarrollo, lo que en algunos casos puede hacer la incorporación inviable. Un tercer grupo está formado por los autores que proponen el uso de patrones para incorporar características de usabilidad en el sistema. Estas propuestas tienen como inconveniente que no especifican los cambios que se deben realizar en la implementación para

dar soporte a la funcionalidad de los patrones. Por último, el grupo de autores que trabaja sobre cómo incorporar características de usabilidad en un proceso de desarrollo MDD es el que menos propuestas presenta.

Concretamente, se han detectado dos trabajos relevantes en el contexto MDD: [Abrahao, 2006] [Abrahao, 2005] [España, 2006]. y [Tao, 2005]. Sin embargo, estos trabajos no definen un método para incorporar aspectos de usabilidad en cualquier entorno MDD, sino que describen de manera generalista cómo tratar la usabilidad en este tipo de entornos sin entrar a profundizar sobre cómo modelar los atributos de usabilidad en un modelo conceptual. Es decir, no describen cómo usar primitivas conceptuales para representar la usabilidad en un entorno MDD concreto, tal y como propone esta tesis. Por tanto, se puede afirmar que la principal contribución de la tesis, el desarrollo de un método para incorporar mecanismos de usabilidad en un entorno MDD, es un trabajo novedoso.

Por otro lado, otra de las contribuciones de la tesis es la aplicación del método para incorporar los mecanismos de usabilidad a una tecnología MDD específica, como es OO-Method. No se han encontrado en la literatura trabajos que hayan aplicado de manera tan práctica su propuesta sobre una herramienta MDD concreta. En la tesis se aborda en detalle los cambios que habría que aplicar a OO-Method, tanto a nivel de modelado conceptual como de compilador de modelos, para incorporar los mecanismos de usabilidad.

Por último, otra de las contribuciones de la tesis es la de evaluar de manera experimental si la incorporación de los mecanismos de usabilidad mejora la usabilidad del sistema o no. Los resultados de esta evaluación serán válidos no sólo para entornos MDD, sino para cualquier entorno de desarrollo que utilice los mecanismos de usabilidad de Juristo et al. Aunque existen varios trabajos que miden la complejidad de tratar con los mecanismos de usabilidad en el proceso de desarrollo tradicional ([Juristo, 2007, a]), son pocos los que miden si la usabilidad del sistema mejora o no con dichos mecanismos. Por lo tanto, este estudio es un trabajo novedoso y de alto valor para la comunidad científica, ya que el hecho de que los resultados sean favorables, justifica el trabajo de incorporar los mecanismos de usabilidad dentro del proceso de desarrollo.

3.3 Aproximación a la solución

Esta sección explica la solución que se plantea al problema anteriormente comentado: cómo incorporar la usabilidad en un proceso de desarrollo de software dirigido por modelos. Esta sección se divide en dos subsecciones:

1. La primera subsección explica los FUFs y cómo estos se dividen en mecanismos de usabilidad, utilizados como base para la propuesta del trabajo de investigación.
2. La segunda subsección explica en detalle el tipo de entorno de desarrollo sobre el que se va a aplicar la propuesta (entornos MDD)

3.3.1 Funcionalidades de usabilidad

A la hora de hacer una propuesta para modelar la usabilidad es necesario tomar una lista de características de usabilidad que serán las que se representarán en el modelo conceptual. De entre todas las propuestas existentes, se han seleccionado las características de usabilidad funcionales descritas en los trabajos de Juristo [Juristo, 2007, b]. Este autor define un conjunto de características de usabilidad que tienen alguna relación con requisitos funcionales. A estas características se les conoce con el nombre de **Functional Usability Features (FUF)**. En primer lugar, esta sección va a explicar el proyecto dentro del cual se definieron los FUFs. En segundo lugar, se presentan los mecanismos de usabilidad en los que se dividen los FUFs y la propuesta de Juristo para incorporar dichos mecanismos en el proceso de desarrollo. Por último, se explica porqué se han elegido los FUFs de Juristo y no otros de los existentes en la literatura.

3.3.1.1 El proyecto STATUS

Tal como se comentó en el estado de la cuestión, la lista de FUFs se ha elaborado en base a los trabajos de Bass [Bass, 2003] y sobre todo a partir de un proyecto europeo llamado STATUS [STATUS, 2008]. En este proyecto se estudiaron las implicaciones que tiene en la arquitectura la incorporación de aspectos de usabilidad que incluyen funcionalidad. Para detectar dichas implicaciones, el proyecto estuvo centrado en el estudio de tres elementos de usabilidad:

- **Atributos de usabilidad:** Por atributo de usabilidad se entiende aquel componente medible del concepto abstracto que es usabilidad. Los atributos de usabilidad identificados por Juristo son:
 - **Facilidad de aprendizaje:** Mide cuánto de sencillo y rápido puede el usuario empezar a ser productivo en un sistema que es nuevo para él. Está relacionado con la facilidad de recordar cómo funciona el sistema.
 - **Eficiencia de uso:** El número de tareas por unidad de tiempo que el usuario puede realizar con el sistema.
 - **Fiabilidad:** Se refiere al ratio de error que el usuario puede tener al utilizar el sistema.
 - **Satisfacción:** Son las opiniones subjetivas que se crea el usuario al utilizar el sistema.

Muchas de las propuestas de la comunidad IPO se quedan a este nivel de detalle para describir cómo construir sistemas usables. Sin embargo, a este nivel de descripción es muy difícil detectar requisitos de usabilidad que guíen la construcción de la arquitectura del sistema. Por lo tanto, son necesarios requisitos de usabilidad para influir en la arquitectura a partir del espacio de la captura de requisitos.

- **Propiedades de usabilidad:** Estas propiedades agrupan los heurísticos y principios de diseño que los investigadores en usabilidad han detectado que tienen una gran influencia en la usabilidad del sistema. En el proyecto STATUS se clasificaron las propiedades de usabilidad en:
 - **Retroalimentación:** El sistema proporciona retroalimentación sobre las acciones que ejecuta.
 - **Gestión de errores:** Incluye la prevención de errores y la recuperación a partir de un error.

- Consistencia: Consistencia entre la interfaz de usuario y las acciones funcionales del sistema.
- Orientación: Guiar al usuario en todo momento.
- Minimizar la carga cognitiva: El diseño del sistema debería reconocer limitaciones cognitivas del usuario.
- Correspondencia natural: Incluye la predicción de acciones, significación semiótica de símbolos y facilidad de navegación.
- Accesibilidad: Incluye acceso multimodal e internacionalización.

Estas propiedades de usabilidad capturan los requisitos de usabilidad de una forma más directa que los atributos, se pueden tratar en la fase de diseño y tienen implicaciones en la construcción del sistema.

- **Patrones de usabilidad:** Con este término se refiere a una técnica o mecanismo que se puede aplicar al diseño de la arquitectura de un sistema para satisfacer una necesidad de usabilidad. Cada uno de los patrones se define con los siguientes elementos:
 - Nombre: Indicador de su propósito.
 - Descripción: Una breve descripción del mecanismo o técnica.
 - Relación con la arquitectura: Impacto que tiene en la arquitectura.
 - Relación con propiedades de usabilidad: Identifica las propiedades de usabilidad relacionadas con el patrón.
 - Ejemplo: Un ejemplo de uso del patrón en un sistema.

La Figura 3.1 muestra de forma gráfica la relación existente entre atributos, propiedades y patrones de usabilidad. Las propiedades de usabilidad

forman parte de los requisitos que dirigen la construcción de la arquitectura. Para la construcción de la arquitectura se deben seleccionar los patrones que satisfacen las propiedades de usabilidad. El proceso de construcción de la arquitectura es un proceso iterativo donde cada modificación de la arquitectura se evalúa. Una vez los componentes han sido diseñados e implementados, el sistema resultante se puede evaluar y medir con respecto a los atributos de usabilidad. Si los resultados de esta evaluación muestran que la usabilidad no es la adecuada, se debe rediseñar la arquitectura del sistema.

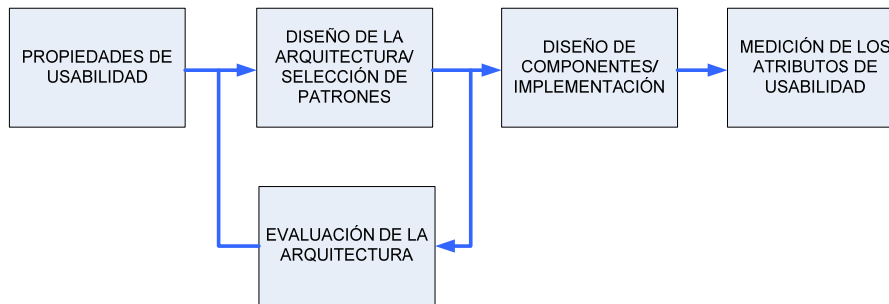


Figura 3.1 Relación entre atributos, propiedades y patrones de usabilidad

Uno de los objetivos que se pretendía alcanzar con el proyecto STATUS era el de minimizar el rediseño de la arquitectura. Para ello, en el ámbito del proyecto se establecieron relaciones entre propiedades, atributos y patrones de usabilidad. De esta forma, el analista puede saber qué propiedades, y a su vez qué atributos de usabilidad, se ven afectados por la incorporación a la arquitectura de cada uno de los patrones de usabilidad. Por ejemplo, el patrón de usabilidad *Asistente* está relacionado con la propiedad *Orientación* que a su vez está relacionada con el atributo *Facilidad de aprendizaje*. A continuación se presentan las funcionalidades de usabilidad que se han detectado a partir del proyecto STATUS.

3.3.1.2 Mecanismos de usabilidad

Partiendo de atributos, propiedades y patrones definidos dentro del proyecto STATUS, Juristo definió los FUFs [Juristo, 2007, b] como recomendaciones para mejorar la usabilidad del sistema que están relacionados con requisitos funcionales. Los requisitos que guían la configuración de los FUFs se consideran requisitos funcionales ya que no sólo tienen implicaciones en la

interfaz, sino también en la funcionalidad. El objetivo que se persigue con la definición de los FUFs es el mismo que el objetivo que se pretendía con el proyecto STATUS, es decir, tratar la usabilidad desde las primeras etapas del proceso de desarrollo software. La principal ventaja de esta aproximación es que la usabilidad se tiene en cuenta en la fase de construcción de la arquitectura y se evitan modificaciones de la misma una vez se haya representado toda la funcionalidad de la lógica de negocio.

La lista de FUFs definidos por Juristo es la siguiente:

- **Feedback:** El objetivo de este FUF es el de mantener informado al usuario de qué es lo que está ocurriendo dentro del sistema.
- **Undo/Cancel:** Permite al usuario cancelar una acción o volver a un estado anterior al cual se encuentra.
- **User input errors prevention/Correction:** Facilitar la entrada de datos de los usuarios de forma que sea lo más rápida posible.
- **Wizard:** Ayuda a los usuarios a ejecutar acciones que requieren más de un paso.
- **User profile:** Permite al usuario adaptar el sistema a sus preferencias.
- **Help:** Proporciona ayuda al usuario sobre cómo llevar a cabo las acciones del sistema.
- **Command aggregation:** Ayuda al usuario a lanzar la ejecución de más de una acción mediante el uso de comandos.
- **Shortcuts:** Permite a los usuarios lanzar una acción de forma rápida.
- **Reuse information:** Permite mover fácilmente información de una parte del sistema a otra parte.

La comunidad IPO considera que a partir de este nivel de descripción de los FUFs, el analista ya debe ser capaz de llevar a cabo la correcta

implementación de la funcionalidad escogida por el usuario. Sin embargo, existen distintas formas de conseguir cada uno de los objetivos planteados en los FUFs. Por ejemplo, el objetivo del FUF *Feedback* se puede conseguir informando al usuario cada vez que realiza una acción, antes de realizar una acción o indistintamente realice o no alguna acción. El objetivo de cada una de estas situaciones es el mismo, pero su contexto o ámbito de aplicación no. Por lo tanto, el analista necesita de más información que la proporcionada en la definición de los FUFs para el desarrollo de sistemas; es esto lo que se conoce como **mecanismos de usabilidad**. Los mecanismos de usabilidad son subtipos de cada FUF. Es decir, cada FUF tiene un objetivo general que se puede especializar en objetivos más detallados llamados mecanismos de usabilidad. Los mecanismos de usabilidad, agrupados por FUF, son los que se muestran en la Tabla 3.1.

FUF	Mecanismo de Usabilidad	Objetivo del Mecanismo de Usabilidad
Feedback	System Status	Informar al usuario sobre el estado interno del sistema
	Interaction	Informar al usuario que el sistema ha detectado una petición de interacción por parte del usuario
	Warning	Informar al usuario sobre acciones que pueden tener consecuencias importantes
	Progress	Informar al usuario que el sistema está procesando una acción que requiere más tiempo para su ejecución
Undo/Cancel	Global Undo	Deshacer acciones del sistema en diferentes niveles

FUF	Mecanismo de Usabilidad	Objetivo del Mecanismo de Usabilidad
	Object-Specific Undo	Deshacer varias acciones de un objeto
	Abort Operation	Cancelar la ejecución de una acción o de toda la aplicación
	Go Back	Volver a un estado anterior en la secuencia de ejecución
User Input Error Prevention/ Correction	Structured Text Entry	Ayudar a prevenir al usuario de la entrada de datos erróneos
Wizard	Step by Step	Ayudar al usuario a ejecutar acciones que requieren varios pasos y varias entradas de datos por parte del usuario
User Profile	Preferences	Guardar las opciones de usuario que oferta la funcionalidad del sistema
	Personal Object Space	Guardar las opciones de usuario que ofertan las interfaces del sistema
	Favourites	Guardar ciertos lugares de interés para el usuario
Help	Multilevel Help	Proporcionar diferentes niveles de ayuda para diferentes usuarios
Command Aggregation	Command Aggregation	Dar la posibilidad al usuario de ejecutar acciones en base a comandos más cortos (agregados)

Tabla 3.1 División de FUF en mecanismos de usabilidad [Juristo, 2007, b]

Con el objetivo de capturar los requisitos que configuren de forma detallada cada uno de los mecanismos de usabilidad, Juristo ha definido una serie de guías. Estas guías están especialmente pensadas para la captura de requisitos de usabilidad con implicaciones funcionales. Las guías ayudan al analista a conocer las implicaciones que tienen en el sistema cada uno de los mecanismos y sobre cómo elicitar estos requisitos. El analista debe orientar las preguntas que realice al usuario para la captura de requisitos de usabilidad en base a estas guías. Las guías para la captura de requisitos de usabilidad se elaboraron en base a trabajos de la comunidad IPO, de entre los cuales destacan los de [Griffiths, 2002][Welie, 2000][Tidwell, 2005][Coram, 1996]. Estas guías se pueden encontrar en el Anexo I.

Las guías para la captura de requisitos son entrada para el siguiente elemento de la propuesta de Juristo, los patrones de usabilidad. Los *patrones de usabilidad* se utilizan para abstraer la forma en la que los analistas pueden incluir los FUFs dentro del proceso de desarrollo del sistema. Cada uno de los patrones de usabilidad está definido con los siguientes apartados:

- **Identificación:** Identifica el mecanismo de usabilidad representado en el patrón.
- **Problema:** Describe el problema de usabilidad que intenta resolver el patrón. Es decir, cómo elicitar y especificar la información necesaria para incorporar el mecanismo de usabilidad en el sistema.
- **Contexto:** Describe el contexto en el que el patrón es útil.
- **Solución:** Es la solución propuesta por el patrón para resolver el problema de usabilidad. La solución está compuesta por dos elementos. Por un lado, *la guía de elicitación del mecanismo de usabilidad* proporciona conocimiento para elicitar la información referente al mecanismo de usabilidad. Esta guía lista las cuestiones que se deben discutir con el usuario para definir la funcionalidad del mecanismo de usabilidad conforme a los requisitos del usuario. Por otro lado, *la guía de especificación del mecanismo de usabilidad* proporciona un ejemplo de esqueleto de especificación.

- **Los patrones relacionados:** Son los patrones que se refieren a otros patrones de usabilidad cuyo contenido esté relacionado con el patrón que se describe.

La descripción de estos patrones va acompañada de Diagramas de Clases y de Secuencia donde se propone cómo incorporar estos patrones a las clases del sistema que dan soporte a la lógica de negocio.

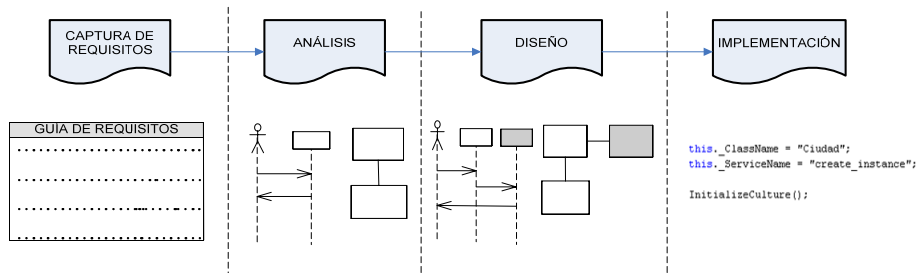


Figura 3.2 Proceso de incorporación de mecanismos de usabilidad a los sistemas propuesto por Juristo

En base a los mecanismos de usabilidad, las guías para la captura de sus requisitos y los patrones que recogen cómo incorporarlos en la arquitectura, Juristo [Juristo, 2007, b] ha definido un método para incorporar dichos mecanismos dentro de un proceso de desarrollo software. En la Figura 3.2 hemos visualizado ese método. El proceso comienza en la fase de captura de requisitos, donde el analista, además de capturar los requisitos para la lógica de negocio del sistema, sigue las guías proporcionadas para extraer los requisitos correspondientes a los mecanismos de usabilidad. Estas guías se componen de una serie de plantillas con preguntas que el analista debe formular al usuario para detectar los requisitos de los mecanismos de usabilidad. Una vez obtenida esa información, el analista debe modelar los mecanismos de usabilidad en la fase de análisis y diseño como si de un requisito más se tratara. En la fase de análisis se modela de forma abstracta los requisitos capturados mediante las guías. Posteriormente, en la fase de diseño, se proporciona el detalle suficiente para incorporar la funcionalidad de los mecanismos de usabilidad dentro a una plataforma específica. Es en estas dos fases donde cada mecanismo de usabilidad queda incorporado a la arquitectura y a la lógica de negocio del sistema. Por último, en base a

estos modelos, se implementa la funcionalidad de los mecanismos de usabilidad junto con la funcionalidad que implementa la lógica de negocio.

3.3.1.3 Ventajas de los mecanismos de usabilidad con respecto a otras propuestas

El método que propone esta tesis para incorporar la usabilidad dentro de un proceso de desarrollo software MDD está basado en los FUFs definidos por Juristo et al [Juristo, 2007, b]. Esta elección está sustentada en las ventajas que ofrecen los FUFs con respecto al resto de guías de usabilidad y heurísticos definidos por la comunidad IPO. Estas ventajas son:

- **Tratamiento de la usabilidad desde la fase de captura de requisitos:** Tal y como hemos explicado en el capítulo Estado de la Cuestión, son varias las propuestas existentes en la literatura que tratan la usabilidad en los requisitos. Sin embargo, la mayoría de las propuestas están centradas en la captura de requisitos de usabilidad no funcionales, olvidando aquellas características de usabilidad que contienen aspectos funcionales. Esto se debe a que consideran la usabilidad como un requisito no funcional.
- **Los requisitos guían la construcción de la arquitectura:** Esta ventaja se deriva de la anterior. Los requisitos de usabilidad que ayudan a configurar los FUFs se pueden considerar requisitos funcionales, por lo que la arquitectura del sistema se define en base a estos requisitos junto con los requisitos que capturan la lógica de negocio del sistema. De esta forma se evitan modificaciones de la arquitectura una vez se haya finalizado su construcción porque desde primer momento se tiene en cuenta toda la funcionalidad a la que debe dar soporte.
- **Las guías para la captura de requisitos están muy detalladas:** Las guías definidas por Juristo para la captura de requisitos de usabilidad están muy detalladas, hasta el punto de incluir las preguntas que el analista debe formular al usuario a fin de capturar los requisitos de forma adecuada. Este nivel de detalle nos ha permitido conocer qué alternativas de configuración tiene cada uno de los mecanismos de usabilidad, ya que es necesario conocer qué

alternativas puede tener el analista en su modelo conceptual para modelar la usabilidad.

- **La descripción de los patrones da una idea de cómo implementar los mecanismos de usabilidad:** En la propuesta de Juristo no sólo se detalla cómo capturar los requisitos, sino que también propone cómo engarzar los mecanismos de usabilidad en la arquitectura del sistema en base a unos patrones de usabilidad. Estos patrones van acompañados de Diagramas de Clases y Secuencia que ilustran cómo las clases que implementan la funcionalidad del mecanismos de usabilidad están relacionadas con el resto de clases del sistema. Esta información es útil para deducir una estrategia de generación de código que implemente los mecanismos de usabilidad.
- **Las guías de captura de requisitos y los patrones de usabilidad son abstractos:** Aunque la definición de las guías de captura de requisitos y de los patrones de usabilidad está muy bien detallada, ambas definiciones se han hecho de forma abstracta para que sirvan en cualquier método de desarrollo y con cualquier lenguaje de programación. Esto permite que incluso sean aplicables a métodos de desarrollo alejados de los métodos de desarrollo manuales, como son los métodos MDD.
- **Los mecanismos de usabilidad están evaluados empíricamente:** Se han llevado a cabo estudios empíricos para evaluar si los mecanismos de usabilidad mejoran o no la usabilidad del sistema [Juristo et al, 2007, b]. Estos estudios consistían en pruebas con varios grupos de estudiantes, donde cada grupo interactuaba con sistemas que incorporaban distinta cantidad de mecanismos de usabilidad (desde ningún mecanismo a un alto porcentaje de ellos). Este estudio reflejó que los usuarios percibían como aplicaciones más usables aquellas que incorporaban mayor número de mecanismos de usabilidad. Además, este estudio sirvió para determinar el porcentaje de funcionalidad nueva que añade cada uno de los mecanismos de usabilidad; la dificultad que conlleva la incorporación de cada uno de los mecanismos de

usabilidad y los errores más comunes que cometen los analistas a la hora de incorporar mecanismos de usabilidad.

3.4 Tecnologías de transformación de modelos

A lo largo de la historia de la Ingeniería del Software se ha ido aumentando el nivel de abstracción para representar los sistemas software. Al principio los sistemas se implementaban en ensamblador, un lenguaje muy cercano al hardware. Posteriormente se abstrajo hasta los lenguajes de programación. Aunque el salto de abstracción fue muy grande, los primeros lenguajes de programación eran estructurados y las herramientas en la que se escribía dicho código eran poco usables. Como ejemplo de estos lenguajes se encuentran entre otros FORTRAN y Pascal. Posteriormente, se siguieron abstrayendo los lenguajes de programación hasta conseguir lenguajes orientados a objetos, donde se podía representar el mundo real de forma más próxima a los mecanismos cognitivos humanos y donde las herramientas de implementación eran gráficas, facilitando enormemente la labor del analista.

En la actualidad, siguiendo con la tendencia de abstraer el problema, han surgido propuestas para generar sistemas software a partir de modelos siguiendo el paradigma *Model-Driven Development* (MDD) [Mellor, 2003]. El desarrollo de sistemas dirigido por modelos facilita la labor del analista, que sólo debe modelar el futuro sistema de forma abstracta. A partir de estos modelos se puede derivar en una fase posterior el código que implemente dichos modelos mediante una serie de reglas de transformación.

El desarrollo MDD se define formalmente según [Mellor, 2003] como el *concepto a través del cual podemos construir el modelo de un sistema, el cual podemos transformar en algo real*. Por **modelo o vista** se entiende un conjunto de elementos formales que describen algo construido para un propósito concreto. Cada uno de los modelos representa un aspecto distinto del sistema. Por ejemplo, puede haber un modelo para representar la persistencia, otro para representar la funcionalidad, otro para representar la interfaz, etc.

Los modelos están compuestos por un conjunto de **primitivas conceptuales**, que son los “ladrillos” o componentes básicos utilizados para su construcción. Más formalmente, se pueden definir las primitivas conceptuales como un elemento del lenguaje de modelado que permite representar de forma abstracta algún aspecto del sistema. Por ejemplo, en un Diagrama de Clases, la clase es la primitiva conceptual principal, pero también son primitivas conceptuales cada uno de los atributos y los servicios de la clase.

Las primitivas conceptuales están agrupadas por modelos o vistas. A su vez, el conjunto de modelos forman el **modelo conceptual**. El modelo conceptual se utiliza para representar las actividades que elicitán y describen el conocimiento general que un sistema de información particular debe ofrecer. Es decir, un modelo conceptual es una manera de visualizar dominios de forma particular [Olivé, 2007].

La Figura 3.3 muestra un esquema de cómo está compuesto el modelo conceptual. El modelo conceptual está compuesto por varios modelos y cada modelo representa una vista diferente del sistema. A su vez, cada modelo utiliza primitivas conceptuales para representar un aspecto del sistema.

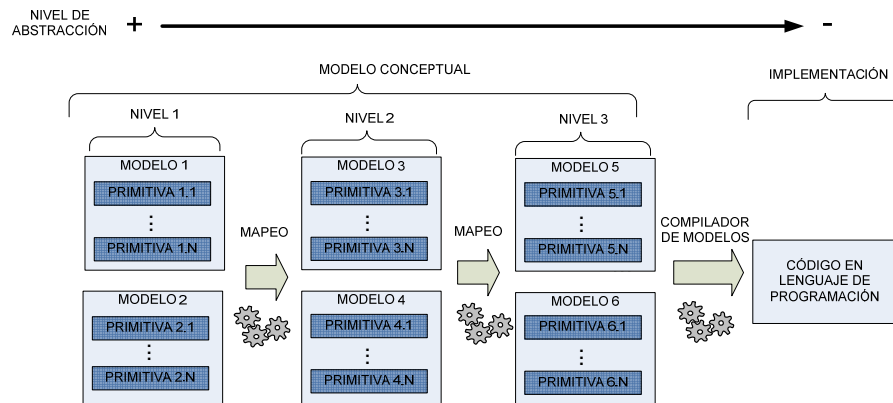


Figura 3.3 Vista esquemática de los componentes de un modelo conceptual

Un método de desarrollo MDD consiste en la representación de un sistema en varios niveles de abstracción. El analista debe especificar una serie de

aspectos del sistema en cada uno de estos niveles. En los métodos de desarrollo MDD, el nivel donde se representa el sistema de forma abstracta se conoce con el nombre de **espacio del problema**. En cambio, el nivel que representa el sistema ya implementado se conoce con el nombre de **espacio de la solución**. Siguiendo los consejos del desarrollo MDD, el analista debe dedicar el mayor esfuerzo posible al modelado del sistema en el espacio del problema, dejando la construcción del espacio de la solución a las transformaciones entre modelos, que serán tan automáticas como sea posible.

Los modelos que componen el espacio del problema no son cajas cerradas e independientes entre sí, sino que tienen relaciones que permiten la transformación de un modelo en otro de un nivel menos abstracto, es decir, del espacio del problema al espacio de la solución. A estas relaciones se les conoce con el nombre de **correspondencias**, y hacen posible que en cada nivel de abstracción se puedan expresar diferentes aspectos del sistema, manteniendo los modelos de cada nivel sincronizados entre sí. De esta forma, cada cambio que el analista realice en el nivel más abstracto, se debe ver reflejado en los niveles menos abstractos. Por ejemplo, lo representado en el modelo que representa la persistencia del sistema puede generar parte del modelo que representa la interfaz. Esta sincronización entre los distintos niveles de abstracción se conoce con el nombre de **trazabilidad**. Uno de los aspectos más importantes que debe tener en cuenta la trazabilidad es cómo mantener la corrección de los distintos niveles de abstracción tras sucesivas modificaciones de los niveles más abstractos. Para garantizar la trazabilidad es conveniente utilizar transformaciones automáticas para asegurar que cada vez que el analista modifique cualquier primitiva conceptual, se refleje este cambio en los modelos con un nivel de abstracción menor. La trazabilidad facilita la labor del analista, que sólo debe preocuparse de modelar ciertas primitivas conceptuales, dejando las actualizaciones de los modelos menos abstractos a las correspondencias.

La transformación del modelo conceptual a código de un lenguaje de programación se lleva a cabo de forma similar a la transformación que un compilador realiza de un lenguaje de programación a código máquina. Debido a este símil, las transformaciones del modelo conceptual a código se conocen con el término de **compilación de modelos**. Por lo tanto, al

responsable de aplicar las transformaciones entre estos dos niveles de abstracción se le suele denominar **compilador de modelos**. Es decir, el compilador de modelos aplica las correspondencias necesarias para generar el código final que implementa el sistema a partir del modelo conceptual.

La Figura 3.3 muestra de forma gráfica la trazabilidad entre los distintos niveles de abstracción. En la figura, el *nivel 1* equivale al nivel más abstracto, mientras que el *nivel 3* corresponde al menos abstracto. En un mismo nivel puede haber varios modelos, por ejemplo, el *modelo 1* y *2* están en el mismo nivel de abstracción. El paso de un nivel de abstracción a otro inferior se realiza en base a correspondencias, representadas en la figura mediante flechas. La trazabilidad asegura que todos los cambios realizados en los niveles pertenecientes al espacio del problema se propaguen hacia niveles inferiores. El compilador de modelos es el encargado de generar el código a partir del conjunto de modelos que componen el modelo conceptual.

Para que el compilador de modelos pueda generar el código a partir del modelo conceptual, se debe utilizar un lenguaje formal y no ambiguo para construir los modelos. Para que un lenguaje sea formal y no ambiguo, debe tener una sintaxis y una semántica bien definidas.

- **Sintaxis:** Define cómo se deben unir las distintas Primitivas Conceptuales para formar un modelo conceptual. La sintaxis puede ser tanto gráfica como textual.
- **Semántica:** Es el significado de lo que hay representado en el modelo conceptual con una sintaxis correcta.

Un diagrama con cajas y líneas entre las cajas que no tiene un significado para esas cajas y líneas no es un modelo, es simplemente un diagrama no formal. Por lo tanto, un modelo MDD debe tener una definición para su parte sintáctica y semántica.

El que los modelos se representen de forma no ambigua permite llevar a cabo las transformaciones entre modelos. Para ello hay que definir las posibles transformaciones entre modelos o correspondencias.

Normalmente, estas correspondencias se definen utilizando un metamodelo. Un *metamodelo* se define como un modelo con la capacidad de almacenar diferentes modelos. En MDD, los metamodelos se suelen representar con la notación QVT [MOF QVT, 2008]. QVT es un lenguaje propuesto por la *Object Management Group (OMG)* que permite la definición de correspondencias entre metamodelos. Este lenguaje permite la definición de un esquema para la consulta, visualización y transformación de metamodelos representados en *Meta-Object Facility (MOF)*. Esta aproximación da lugar a modelos que no son universales, ya que dependen de la instanciación de cada uno de los metamodelos. La Figura 3.4 presenta de forma gráfica cómo un metamodelo puede especificar las transformaciones que se deben dar para convertir un *modelo de entrada* en un *modelo de salida*. Las transformaciones se definen en base a un *metamodelo origen* y a un *metamodelo destino* utilizando un lenguaje no ambiguo.

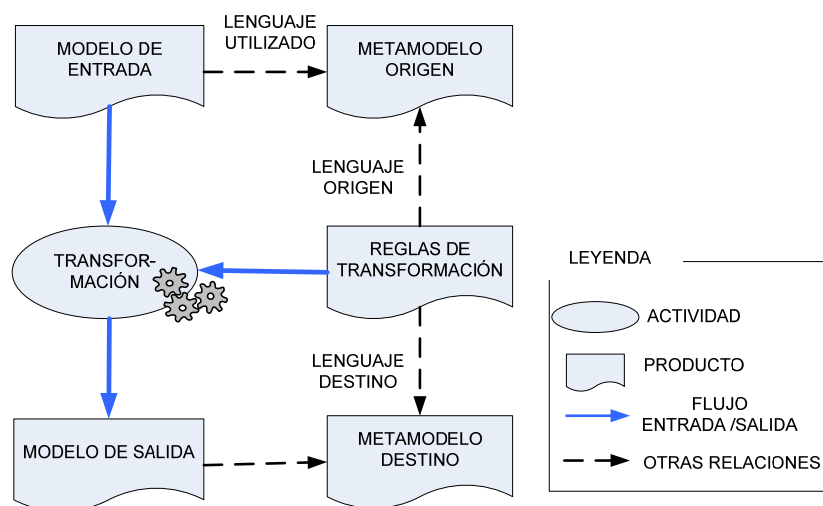


Figura 3.4 Transformaciones expresadas con un metamodelo

Además de proponer la notación QVT, la *OMG* ha propuesto un estándar MDD llamado *Model Driven Architecture (MDA)* [MDA, 2008]. MDA es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos. La estandarización es un paso adelante en la implantación de MDD en el desarrollo de sistemas, porque facilita el reuso y permite el uso de distintas herramientas que dan soporte

al mismo estándar. Tal y como se puede apreciar en la Figura 3.5, los modelos que componen MDA son cuatro:

- **Computation Independent Model (CIM):** Este modelo representa el entorno y los requisitos del sistema de forma totalmente independiente de cualquier soporte tecnológico.
- **Platform Independent Model (PIM):** En este modelo se presentan las operaciones del sistema que se mantienen constantes para cualquier plataforma tecnológica.
- **Platform Specific Model (PSM):** El PIM se transforma en uno o varios PSM. El PSM especifica el sistema en términos de una tecnología de implementación específica.
- **Code Model (CM):** Este modelo representa el código en un lenguaje de programación específico que implementa el sistema.

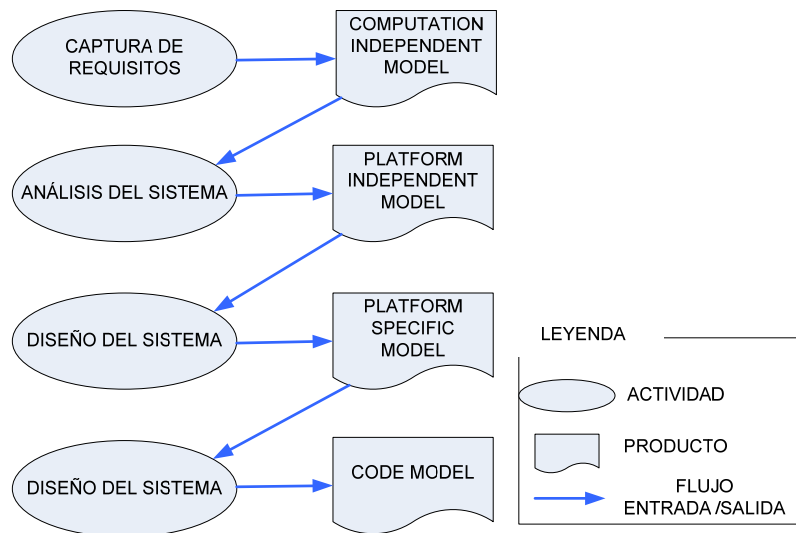


Figura 3.5 Modelos que componen el estándar MDA

La separación que propone MDA en varios modelos que representan el sistema de forma abstracta tiene las siguientes ventajas [Miller, 2001]:

- El nivel PIM permite validar la estructura y el comportamiento de los sistemas de forma totalmente independiente de la plataforma tecnológica sobre la que se implemente.
- El nivel PIM permite expresar una misma estructura y un mismo comportamiento del sistema que se puede implementar en distintas plataformas tecnológicas.
- La integración y la interoperabilidad a través de los sistemas se pueden definir de forma más clara en términos independientes de plataforma. Posteriormente, se pueden proyectar estos términos sobre una plataforma específica en el nivel PSM.

Normalmente, la notación utilizada en MDA para representar los modelos es UML [UML. 2008]. UML proporciona muchas opciones de modelado a los analistas, ya que esta notación puede representar de forma abstracta la estructura y el funcionamiento de cualquier sistema. Para ello, el analista debe utilizar varios modelos UML donde cada uno representa un aspecto del sistema. Los modelos representados con la notación UML son declarativos. UML presenta una serie de ventajas con respecto a otros lenguajes declarativos que hacen que se haya convertido en uno de los más utilizados en el mundo real [Miller, 2001]:

- Los modelos UML pueden ser semánticamente más ricos que los modelos utilizados en otros lenguajes declarativos.
- UML se ha definido formalmente usando conceptos de modelado del propio UML, mostrando su capacidad de representación.
- UML se puede representar tanto de forma gráfica como textual.
- UML permite expresar restricciones del comportamiento de los sistemas de forma precisa mediante el lenguaje *Object Constraint Language (OCL)*. OCL proporciona al analista una serie de instrucciones precisas que dejan poco margen a la suposición. Estas instrucciones son válidas para cualquier tipo de implementación que se lleve a cabo.

Los modelos basados en UML se pueden usar de forma horizontal y vertical.

- **Horizontal:** El modelo describe varios aspectos del sistema en el mismo nivel de abstracción.
- **Vertical:** El modelo describe un aspecto del sistema en varios niveles de abstracción.

Independientemente de la forma en la que se usen los modelos UML, una de las características de cualquier método de desarrollo software basado en MDA es que debe soportar el desarrollo iterativo e incremental del sistema. Esto significa que las correspondencias entre los distintos niveles de abstracción deben ser repetibles, es decir, ser las mismas correspondencias para todos los sistemas desarrollados. Si una correspondencia necesita más información en los modelos origen de la que estos modelos pueden ofrecer, es necesario añadir pequeñas modificaciones en los modelos de origen. Esta información adicional se conoce con el nombre de *anotaciones*. Las correspondencias pueden utilizar estas anotaciones en su proceso de transformación de modelos.

Los pasos necesarios para incorporar MDA en un proceso de desarrollo software se pueden dividir en siete, según [Mellor, 2002]:

1. Identificar las plataformas destino
2. Identificar los metamodelos que se quieren utilizar para describir los modelos de las plataformas seleccionadas. Además, también se debe elegir el lenguaje de modelado en el que se representarán los modelos.
3. Encontrar los metamodelos adecuados que soporten los cambios de plataforma esperados.
4. Definir las técnicas de correspondencia que se usarán en los metamodelos, de forma que haya una ruta desde el metamodelo más abstracto hasta el metamodelo que representa la plataforma destino.
5. Definir los modelos de anotaciones necesarios para las reglas de correspondencia.
6. Implementar las técnicas de correspondencia para que el proceso de transformación entre modelos se pueda automatizar.
7. Dirigir las iteraciones por las que se pasa en el desarrollo del sistema. Cada iteración añadirá detalle a uno o más modelos describiendo el sistema a uno o más niveles de abstracción.

El proceso de correspondencia entre los distintos modelos se puede llevar a cabo de forma manual (el analista), semi-automática (el analista apoyado con alguna herramienta informática) o automática (una herramienta informática). De las tres opciones, la que libera de más trabajo al analista es la última, ya que solo debe crear los modelos y es el propio sistema el que realiza todas las transformaciones de forma automática. Cuando el proceso MDD se automatiza, aparece el concepto de *Tecnología de Transformación de Modelos (TTM)*. Este concepto se define como la capacidad de tomar un modelo abstracto de un sistema y transformarlo de forma automática en otro modelo. Las TTM ofrecen tres características para definir las transformaciones [Sendall, 2003]:

- **Manipulación del modelo:** Es la capacidad de acceder a una representación interna del modelo y de modificarla.
- **Representaciones intermedias:** Los modelos se deben poder representar de alguna forma estándar para poder trabajar con varias herramientas. Uno de los lenguajes estándar muy usado para las representaciones de modelos es XML.
- **Soporte al lenguaje de transformación:** El lenguaje de transformación debe proporcionar una serie de constructores para expresar, componer y aplicar transformaciones.

En la siguiente sección se presentan las ventajas de utilizar una TTM frente a los métodos clásicos (manuales) de desarrollo software. La elección de porqué hemos definido un método para incorporar la usabilidad dentro de una TTM está sustentada en dichas mejoras.

3.4.1 Ventajas e inconvenientes de las tecnologías de transformación de modelos

El uso de modelos para representar sistemas de forma abstracta es una técnica muy extendida dentro del ámbito de la Ingeniería del Software. De la misma forma que hoy en día sería impensable que un arquitecto empezara la construcción de un edificio sin haber realizado previamente los planos de

dicho edificio, también es impensable que los analistas se pongan a implementar un sistema sin previamente haberlo modelado. En muchos casos estos modelos se construyen en las primeras fases de desarrollo y una vez implementado el sistema no se vuelven a utilizar. Los cambios que se producen en las últimas etapas de desarrollo normalmente se hacen directamente sobre el lenguaje de programación que implementa el sistema sin prestar atención a los modelos. Por tanto, los modelos quedan obsoletos, ya que al modificar la implementación y no actualizar el modelo hay una incongruencia entre modelo e implementación. Los modelos que quedan obsoletos pierden toda su utilidad y no son más que documentación sobre cómo se pensó que sería el sistema en las primeras fases del desarrollo.

Con la aparición de MDD, y más concretamente de TTM, los modelos que representan el sistema de forma abstracta son el propio sistema en sí. El compilador de modelos hace posible la transformación de los modelos conceptuales en código de un lenguaje de programación. Por tanto, en una TTM los modelos no se utilizan como documentación del sistema a desarrollar, sino que requieren ser definidos con la máxima precisión porque a partir de estos modelos se generará el sistema. Esta es la nueva filosofía que traen consigo las TTM. A continuación vamos a presentar las ventajas que esta nueva filosofía de desarrollo de sistemas presenta con respecto al desarrollo de sistemas tradicional [Selic, 2003]:

- La principal ventaja de las TTM es que el analista puede expresar el sistema usando conceptos que no están ligados a la tecnología de implementación utilizada. Esto permite especificar el sistema de una forma cercana al dominio del problema sin tener que pensar en aspectos meramente tecnológicos. De esta forma, los modelos son menos sensibles a la tecnología seleccionada para la implementación, consiguiendo que éstos sean más sencillos de especificar, de comprender y de mantener. Es más sencillo para los expertos del dominio el construir sistemas mediante un método TTM que a un experto en lenguajes de programación implementar el sistema.
- El tiempo que el analista debe dedicar a la producción de software es mucho menor que el tiempo que le debería dedicar si

desarrollara el sistema con un método de desarrollo tradicional. Con una TTM, el analista sólo debe construir los modelos conceptuales y es el compilador de modelos el encargado de implementar el sistema de forma totalmente automática. Por lo tanto, la fase de implementación entendida de forma convencional desaparece del proceso de producción de sistemas.

- Si los modelos se convierten en mera documentación pierden todo valor porque la documentación rápidamente diverge del mundo real. Por lo tanto, otra de las ventajas de las TTMs es que los modelos no son la documentación del sistema, sino el propio sistema en sí, ya que el sistema se genera a partir de dichos modelos.
- Los métodos basados en TTMs disponen de lo que se conoce como *round-trip engineering*. Este término es la capacidad de las TTMs de pasar de código en un lenguaje de programación a los modelos conceptuales que representan dicho código de forma abstracta. Es decir, es el proceso contrario al utilizado en el desarrollo de sistemas, pasar del menor nivel de abstracción al mayor. Esta técnica se puede utilizar para detectar errores en el código pero a nivel de modelo conceptual.
- La generación completa de código significa que el compilador de modelos toma el rol de implementador, ya que gracias a él se genera el sistema final tomando como entrada los modelos conceptuales. Este proceso de generación hace posible que los cambios sobre el sistema no se hagan a nivel de código del lenguaje de programación, sino a nivel del modelo conceptual. Estos cambios son mucho más sencillos y requieren menos conocimientos técnicos por parte del analista que si se llevaran a cabo a nivel de código.
- El uso de lenguajes no ambiguos no solo permite automatizar la transformación entre modelos, sino que también permite la verificación de las propiedades que el analista considere oportunas para el sistema. Estos análisis pueden tomar diversas formas, incluyendo la verificación formal o el chequeo simple de algunas de las propiedades de los modelos conceptuales.

- La estrategia de generación de código debe incluir reglas de transformación que generen sistemas de la forma más eficiente posible. Así, los sistemas generados mediante TTM puede que sean más eficientes que los desarrollados por los expertos en la tecnología seleccionada para la implementación.
- En los métodos de desarrollo software tradicionales, muchos errores en el sistema solo se detectan una vez el sistema ya está implementado. Corregir estos errores normalmente supone cambios importantes en el sistema que son muy costosos. Al desarrollar los sistemas con una TTM, el analista puede corregir los errores simplemente modificando los modelos y los cambios en la implementación se delegan a las reglas de transformación entre modelos.
- El desarrollo de sistemas utilizando una TTM puede ser gradual, es decir, el analista puede modelar parte del sistema, generar el sistema que lo implementa y después continuar con el modelado de otras partes del sistema. Todo este proceso se puede hacer utilizando la misma metodología y la misma herramienta de desarrollo para todas las fases.
- Un mismo modelo conceptual se puede utilizar para generar sistemas en varios lenguajes de programación y para ser ejecutados sobre varias plataformas (Web, Escritorio, PDA, etc.). Todo depende de las reglas de transformación que haya dentro del compilador de modelos. Así, el mismo sistema se puede usar tanto para Web, como para escritorio, como para PDA, simplemente seleccionando el sistema destino y dejando que se realice la transformación correspondiente.
- El código generado en un lenguaje de programación está libre de errores sintácticos, siempre asumiendo, por supuesto, que el compilador de modelos funciona correctamente. Los únicos errores que pueden existir serán debidos a un mal modelado por parte del analista. La ausencia de errores sintácticos facilita la labor del analista.

Además de todas estas ventajas, también existen algunos inconvenientes en el uso de métodos de desarrollo basados en TTM. Algunos de estos inconvenientes son [Selic, 2003][Hailpern, 2006]:

- Los compiladores de modelos sólo generan código en base a lo expresado en el modelo conceptual y en la estrategia de generación de código. Es decir, que si el modelo conceptual no tiene la suficiente expresividad como para representar todas las características del sistema, estas características no se podrán generar de forma automática y requerirán cambios manuales en el código generado para poder incorporarlas al sistema.
- En caso de que hayan aspectos del sistema que no se puedan representar, el analista deberá implementarlos a mano, tal y como se ha contado en el punto anterior. Modificar el código generado es una tarea complicada, ya que el analista debe enfrentarse a un sistema que no ha implementado él (solamente lo ha modelado).
- Si se encuentra un error en la ejecución del sistema, puede que sea difícil encontrar la solución en el modelo conceptual. El nivel de dificultad va a depender de las facilidades que ofrezca la herramienta que implementa la TTM. Las herramientas deberían ofrecer una funcionalidad similar a los depuradores de los entornos de desarrollo de tercera generación.
- Los humanos son creativos y en algunos casos pueden optimizar el código de mejor forma de la que ofrece el compilador de modelos. De todas formas, estadísticamente, los compiladores de modelos suelen generar sistemas de forma más eficiente que manual. Aun así, los problemas de eficiencia en el código que implementa el sistema no son una de las mayores preocupaciones de los analistas.
- La existencia de múltiples niveles de abstracción puede desembocar en redundancia a la hora de representar sistemas. Los modelos conceptuales deben tener en cuenta la redundancia en el modelado y eliminarla para agilizar la labor del analista.

- Aquellas TTMs que no soporten el round-trip engineering están expuestas a que aquellos cambios que se realicen en niveles menos abstractos no se vean reflejados en niveles más abstractos. Por ejemplo, los cambios hechos a mano en el código no se verían representados en el modelo conceptual. Esta falta de sincronización puede hacer que al volver a aplicar el compilador de modelos, los cambios manuales se pierdan porque las reglas de transformación solo tienen en cuenta lo expresado mediante el modelo conceptual.
- A mayor número de niveles de abstracción, mayor complejidad para el desarrollo de sistemas. Tal y como aumenta el número de modelos que forman el modelado conceptual, aumentan las relaciones y las correspondencias entre ellos, y por tanto aumenta la complejidad en el desarrollo de sistemas. Es necesario equilibrar la balanza entre el número de modelos abstractos y la facilidad de modelado de la TTM.

Las ventajas de utilizar una TTM son mucho mayores que los inconvenientes que pueden surgir. Además, muchos de estos inconvenientes se pueden resolver modificando ligeramente la TTM, como por ejemplo el caso de incluir el round-trip engineering y ajustar el número de niveles de abstracción a un número óptimo. Por tanto, el uso de las TTMs en el desarrollo de sistemas está justificado en base a estas ventajas. Por este motivo hemos tomado una TTM para la incorporación de características de usabilidad.

3.4.2 Una tecnología de transformación de modelos: OO-Method

De entre todas las TTMs que existen, esta tesis utiliza OO-Method [Pastor, 2007] como ejemplo para incorporar la usabilidad en su modelo conceptual. OO-Method es un método de desarrollo que sigue el estándar MDA, es decir, diferencia el modelado del sistema en diferentes niveles de abstracción. OO-Method está implementado en una herramienta industrial llamada OlivaNOVA [CARE, 2008]. Esta herramienta es capaz de generar código automáticamente a partir de modelos conceptuales OO-Method. Actualmente la herramienta genera código para aplicaciones de escritorio y

aplicaciones Web sobre el lenguaje C# y Java. Por lo tanto, un mismo modelo conceptual se puede utilizar tanto para generar una aplicación de escritorio como para generar una aplicación Web.

A continuación se explica en detalle cada uno de los modelos que conforman los niveles de abstracción de OO-Method y cuáles de ellos están ya incorporados a OlivaNOVA.

3.4.2.1 Computation Independent Model (CIM)

Este nivel de abstracción está cubierto en OO-Method por una serie de modelos que capturan tanto requisitos funcionales como de interacción. Propuestas para la captura de requisitos funcionales existen varias, pero todas ellas están a nivel experimental y aún no han sido incorporadas a OlivaNOVA:

- **Captura de requisitos mediante un Árbol de Refinamiento de Funciones (ARF) [Insfrán, 2002]:** Esta técnica de captura de requisitos consta de tres elementos: (1) la *misión del sistema*: es una descripción de alto nivel de la naturaleza y propósito del sistema. A través de esta definición se debe poder determinar qué hará y qué no hará el sistema; (2) el *árbol de refinamiento de funciones* representa la descomposición jerárquica de las funciones del sistema. El resultado es un árbol donde las hojas representan las funciones del sistema y las raíces representan entradas del menú a partir de las cuales se invocan las funciones; (3) Modelo de Casos de Uso: especifica mediante plantillas y Casos de Uso cada una de las funcionalidades del sistema (las hojas del ARF). Estos Casos de Uso son primarios, es decir, representan las funcionalidades más importantes del sistema.
- **Captura de requisitos funcionales mediante Casos de Uso y plantillas [Díaz, 2003]:** Esta técnica consiste en construir los Casos de Uso, sus plantillas y derivar a partir de ellos el modelo conceptual del sistema. Las transformaciones están basadas en reglas sintácticas del lenguaje en que están escritas las plantillas. Por lo tanto, las plantillas se deben construir en base a unas reglas sintácticas bien definidas.

- **Captura de requisitos de interacción mediante bocetos [Panach, 2007, b]:** Esta técnica está basada en el uso de bocetos para capturar cómo quiere el usuario que sea la interfaz del sistema. Estos bocetos se almacenan, de forma transparente para el analista, en base a una notación de árboles de tareas conocida como *Concur-Task Trees* [Paternò, 2004]. A partir de lo expresado en los árboles de tareas, y con reglas de transformación entre modelos, se puede obtener el modelo abstracto que representa la interfaz del sistema (el Modelo de Interacción Abstracto que se verá a continuación).
- **Captura de requisitos mediante procesos de negocio (BPMN) [De la Vara, 2008]:** Esta propuesta está basada en el uso conjunto de modelado de procesos de negocio de una organización (con la notación BPMN) y análisis de metas de un sistema información (con la aproximación Maps). Tras modelar y analizar procesos de negocios y metas, los requisitos del sistema especifican a partir de las tareas que una organización desea ejecutar para alcanzar sus objetivos, y dichos requisitos son analizados posteriormente para la derivación de modelos de OO-Method.
- **Captura de requisitos mediante análisis comunicacional [España, 2009]:** Esta propuesta se centra en capturar los requisitos del sistema en base a las interacciones comunicacionales que existen entre el sistema y su entorno. El análisis comunicacional está actualmente siendo utilizado en empresas e instituciones públicas.

3.4.2.2 Platform Independent Model (PIM)

Este nivel de abstracción está formado por los modelos que componen el modelo conceptual de OO-Method y a partir de los cuales se genera el código del sistema. El total de modelos que forman el PIM son cuatro, y en cada uno de los modelos se representa una parte distinta del sistema. Todos estos modelos están incluidos en la herramienta OlivaNOVA. Veamos en detalle cada uno de estos modelos:

- **Modelo de Objetos:** Este modelo permite especificar la estructura de clases identificadas en el dominio del problema, así como las

relaciones estructurales y las relaciones de agentes sobre los servicios de las clases. El Modelo de Objetos se representa mediante un Diagrama de Clases en el que se indican los atributos y los servicios para cada una de las clases.

- **Modelo Dinámico:** En este modelo se representan los aspectos del sistema referentes al control, a las posibles secuencias de eventos que pueden ocurrir en la vida de los objetos, y a la interacción entre éstos. Para describir la secuencia posible de eventos se utilizan Diagramas de Transición de Estados (DTE) para cada clase, y para describir la comunicación/interacción entre objetos se utiliza un Diagrama de Interacción.
- **Modelo Funcional:** Este modelo se utiliza para especificar el efecto que tienen los eventos sobre el estado de los objetos. La funcionalidad del sistema se representa dotando a este modelo de un esquema declarativo para recoger toda la información que describe los efectos de los eventos sobre los objetos del sistema.
- **Modelo de Presentación:** En este modelo se especifica la interfaz de usuario y está formado por dos vistas: el *Modelo de Interacción Abstracto* que representa qué elementos se visualizarán en la interfaz, y el *Modelo de Interacción Concreto*, que representa cómo se visualizarán estos elementos en la interfaz. El primer modelo a explicar es el Modelo de Interacción Abstracto. Este modelo se construye mediante la especificación de una serie de patrones de interfaz divididos en tres niveles [Molina, 2003]:
 - Nivel 1, Árbol de Jerarquía de Acciones: Siguiendo el principio de *aproximación gradual* [Barfield, 1993], expresa cómo la funcionalidad será presentada al usuario que accede al sistema.
 - Nivel 2, Unidades de Interacción: Modela las Unidades de Interacción (UI) que el usuario utilizará para llevar a cabo sus tareas. Hay cuatro tipos de UI:
 - **UI de Servicio (UIS):** Modela la presentación de un diálogo cuyo objetivo es que un usuario lance un servicio. El usuario debe proporcionar valor a los

- argumentos del servicio mediante campos de entrada de datos.
- **UI de Población (UIP):** Su objetivo es mostrar un conjunto de instancias para una clase dada.
 - **UI de Instancia (UII):** Modela la presentación de los datos o estado de una instancia.
 - **UI de Maestro/Detalle (UIM):** UI compleja que se construye a partir de UIs más sencillas.
- Nivel 3, Patrones Elementales: Permiten restringir y precisar el comportamiento de las diferentes UIs. Son los siguientes:
- **Introducción:** Captura los aspectos relevantes a los tipos de datos que van a ser introducidos. Este patrón está dentro de una subcategoría de los Patrones Elementales, llamada Patrones Auxiliares.
 - **Selección definida:** Permite enumerar los elementos válidos para un argumento. Este patrón junto con el de Introducción son los únicos que están dentro de la subcategoría Patrones Auxiliares.
 - **Información complementaria:** Captura información adicional para ser mostrada al usuario para complementar el OID del objeto.
 - **Dependencia:** Especifica las reglas de dependencia que pueden existir entre argumentos. Con este patrón se especifica qué argumentos dependen del valor de otros. Estas dependencias se describen mediante la nomenclatura evento-condición-acción. Es decir, al producirse un evento y si la condición es cierta, se desencadena una acción. Los posibles eventos son: SetActive y SetValue. Las posibles acciones son: SetActive, SetValue, SetFocus, SetMandatory y SetVisible.
 - **Recuperación de estado:** Inicializa argumentos en base al objeto sobre el que se ejecuta el servicio.

- **Agrupación de argumentos:** Permite agrupar argumentos cuando su número es elevado.

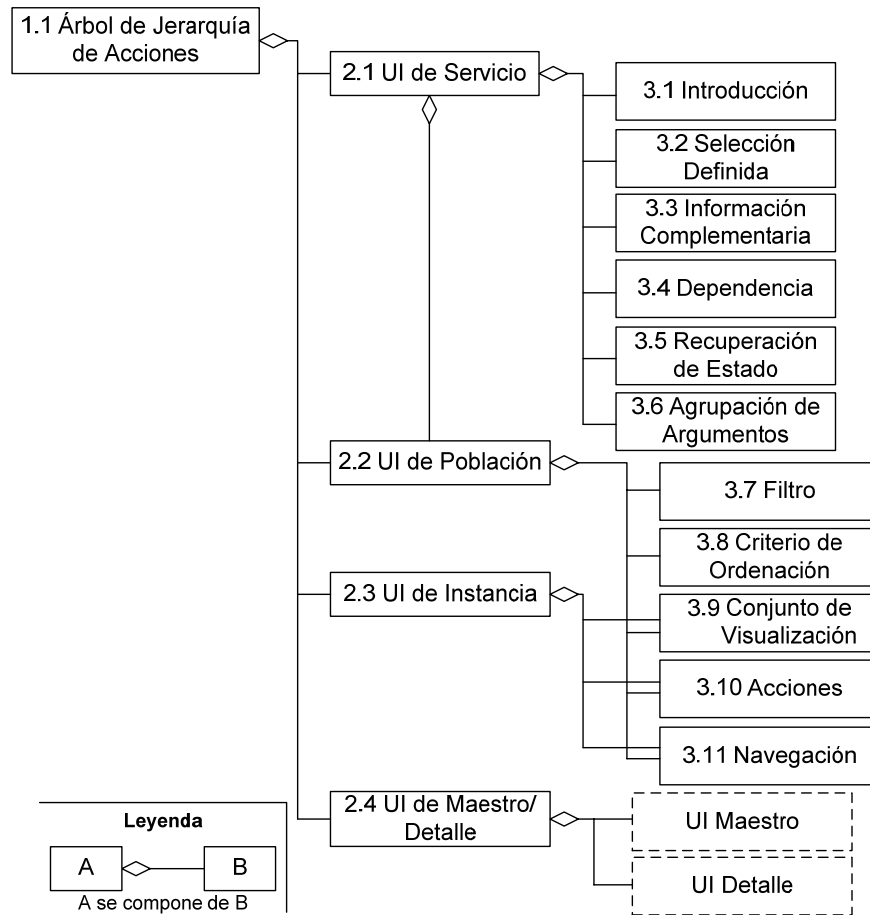


Figura 3.6 Composición de patrones del Modelo de Interacción Abstracto [Molina, 2003]

- **Filtro:** Permite hacer un filtrado de información a partir de un criterio de filtrado.
- **Criterio de ordenación:** Proporciona un mecanismo de ordenación de objetos basado en las propiedades de éstos.
- **Conjunto de visualización:** Describe una lista de propiedades observables (atributos) de los objetos.

- **Acciones:** Determina las acciones que son ofrecidas a un usuario tras seleccionar un objeto.
- **Navegación:** Determina qué información relacionada con el objeto actual es alcanzable en un contexto dado.

La Figura 3.6 muestra de forma gráfica los elementos de los tres niveles y las relaciones entre ellos.

El Modelo de Interacción Abstracto es parte del PIM, pero el Modelo de Interacción Concreto es parte del PSM, ya que se especifica cómo se visualizarán los elementos de la interfaz para una plataforma específica. Este es el motivo por el que el Modelo de Interacción Concreto se explica en el siguiente nivel de abstracción.

3.4.2.3 Platform Specific Model (PSM)

Este nivel de abstracción está formado por el Modelo de Interacción Concreto y el compilador de modelos. Ambos elementos se encuentran en este nivel porque especifican aspectos del sistema que son dependientes de plataforma. OlivaNOVA sólo soporta el compilador de modelos.

- **Modelo de Interacción Concreto** [Aquino, 2008]: Este modelo toma como entrada el Modelo de Interacción Abstracto y especifica cómo se visualizarán los elementos detallados en él. La Figura 3.7 muestra cómo sería el proceso. A lo especificado en el Modelo de Interacción Abstracto se le aplican una serie de plantillas de visualización a través de un editor gráfico. Estas plantillas contienen información sobre las características de visualización de los elementos de las interfaces y aseguran que estas características sean homogéneas en todas las interfaces del sistema. Las plantillas almacenan información relativa al formato y a la distribución de los elementos de la interfaz.
- **Compilador de modelos:** Es el encargado de tomar todos los modelos que forman el modelo conceptual y el Modelo de Interacción Concreto con el objetivo de generar el código que refleje lo expresado en dichos modelos. El código generado por medio del

compilador de modelos tiene tres capas: interfaz, lógica de negocio y persistencia.

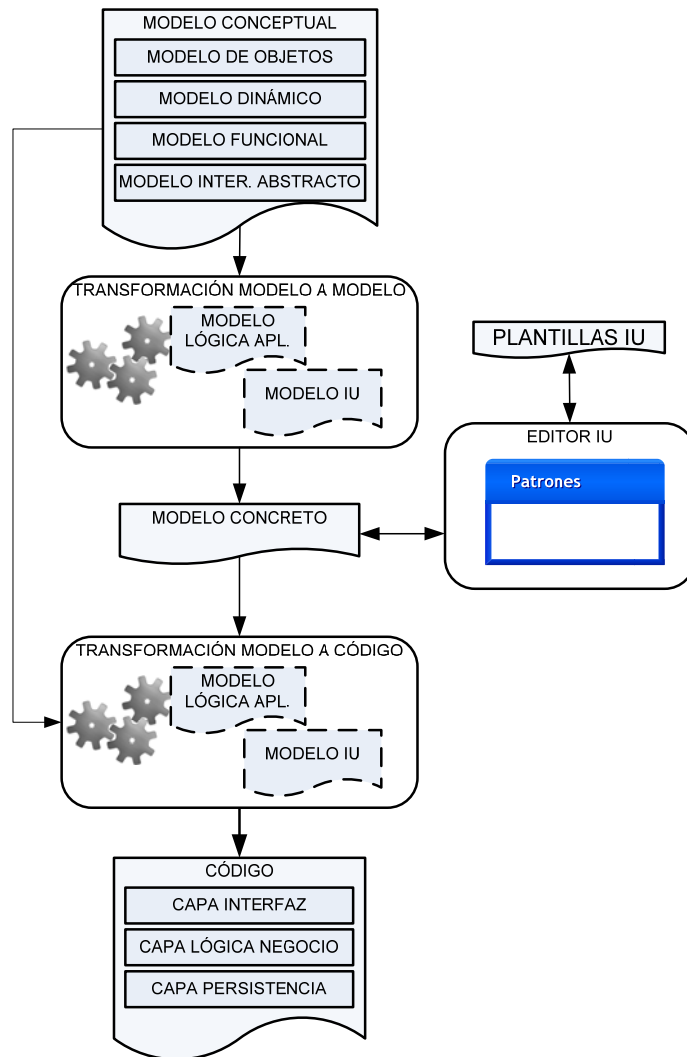


Figura 3.7 Modelo de Interacción Concreto de OO-Method

3.4.2.4 Code Model (CM)

El compilador de modelos genera un código específico para resolver un problema concreto. El código generado será específico para aplicaciones Web o de Escritorio, dependiendo de las necesidades del usuario. Cada una

de las UIs y de los Patrones Elementales dan lugar a una serie de clases con una arquitectura Cliente/Servidor, tal como muestra la Figura 3.8. Este tipo de arquitectura divide la aplicación en dos partes: el cliente y el servidor. La parte servidora es la parte que contiene la capa de la lógica del negocio y la de persistencia. Por otro lado, la parte cliente implementa las interfaces del sistema. La arquitectura Cliente/Servidor tiene la ventaja de que un mismo servidor puede dar servicio a varios clientes. La parte servidora se instala en una máquina a la cual pueden acceder los clientes a través de Internet.

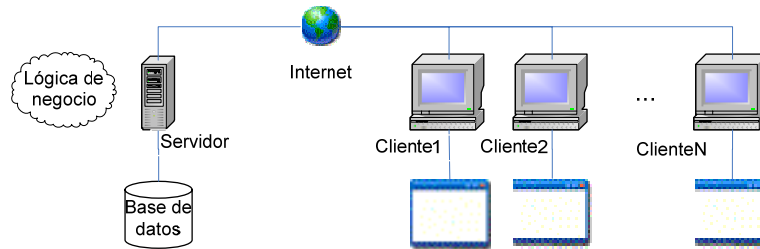


Figura 3.8 Arquitectura Cliente / Servidor de OlivaNOVA

Es importante resaltar que OO-Method está pensado para el desarrollo de aplicaciones de gestión, es decir, aplicaciones que acceden a una base de datos a realizar operaciones con datos almacenados. Las acciones que se hacen sobre la base de datos son inserciones, borrados y modificaciones, dependiendo de la lógica de negocio que implemente el sistema. Este tipo de aplicaciones es uno de los más demandados por la industria, ya que permite almacenar información útil para la gestión del negocio. OO-Method no está pensado para sistemas de ámbitos como la inteligencia artificial, sistemas multimedia, sistemas de tiempo real, etc².

A continuación se detalla mediante Diagramas de Clase, las clases generadas para cada UI tomando el generador de código de C# 2003. Estos

² Las ideas generales de OO-Method podrían ser proyectadas sobre cualquier entorno, siempre que se incorporen en dicho entorno las particularidades y expresividad requeridas por el dominio considerado.

diagramas se han representado con Clases Genéricas, es decir, clases abstractas que no han sido instanciadas para ningún sistema en particular:

- **UI de Servicio:** Cada UI de Servicio modelada en el Modelo de Interacción Abstracto da lugar a un formulario distinto que como nombre incluye el nombre de la UI de Servicio de la que se deriva. Para representar cualquier formulario de este tipo se ha utilizado el término *Form X* (Figura 3.9). Las clases *OK Button* y *Cancel Button* representan los botones para lanzar y cancelar un servicio. La clase *Service Wrapper* llama a las clases del servidor que contienen el código con la lógica de negocio de los servicios que forman el sistema. Es decir, esta clase se encarga de hacer de conector entre las clases del cliente y las del servidor.

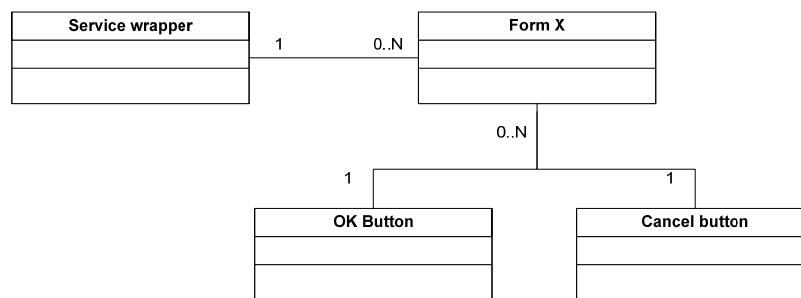


Figura 3.9 Clases generadas a partir de una UI de Servicio

- **UI de Población:** La clase principal que implementa esta UI es *FrmGnPopulation*, tal como muestra la Figura 3.10. Esta clase representa todas las UI de Población modeladas en el Modelo de Interacción Abstracto. Además, esta clase mantiene relaciones con Patrones Elementales que añaden funcionalidad a la UI de Población: filtro (*Filter*); Criterio de Ordenación (*Order Criteria*); Navegación (*Navigation*); Conjunto de Visualización (*Display Set*); Acción (*Action*).

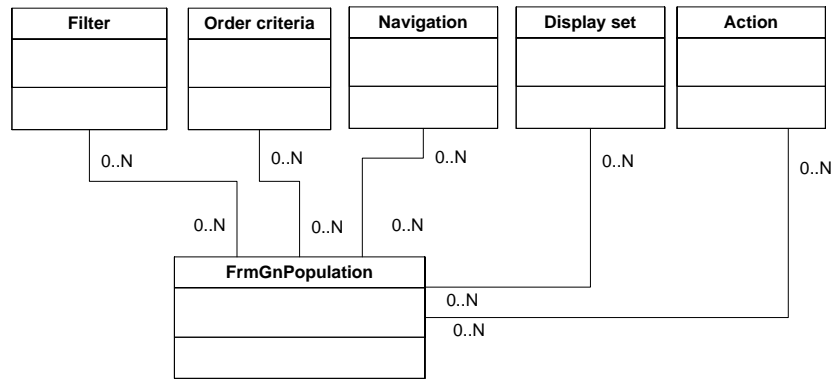


Figura 3.10 Clases generadas a partir de una UI de Población

- UI de Instancia:** La clase principal que implementa esta UI es *FrmGnInstance*, que representa todas las UI de Instancia definidas en el Modelo de Interacción Abstracto, tal y como muestra la Figura 3.11. Además, existen relaciones con Patrones Elementales cuya funcionalidad va ligada a la UI de Instancia: Navegación (*Navigation*); Acción (*Action*) y Conjunto de Visualización (*Display Set*).

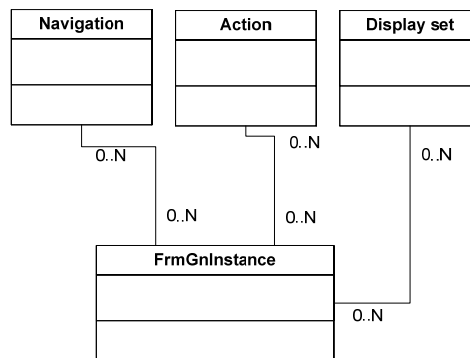


Figura 3.11 Clases generadas a partir de una UI de Instancia

- UI de Maestro/Detalle:** Cada UI de Maestro/Detalle modelada en el Modelo de Interacción Abstracto da lugar a un formulario distinto que como nombre incluye el nombre de la UI de Maestro/Detalle de la que se deriva. Para representar cualquiera de estos formularios se ha utilizado el término *Form X Master* (Figura 3.12). *Form X*

Master se relaciona con las clases *FrmGnInstance* y *FrmGnPopulation*, cuyas instancias se pueden convertir en clases maestra y detalle. Sólo una clase (*FrmGnInstance* o *FrmGnPopulation*) puede ser maestro, pero no hay límite para las clases detalle (tanto *FrmGnInstance* como *FrmGnPopulation*).

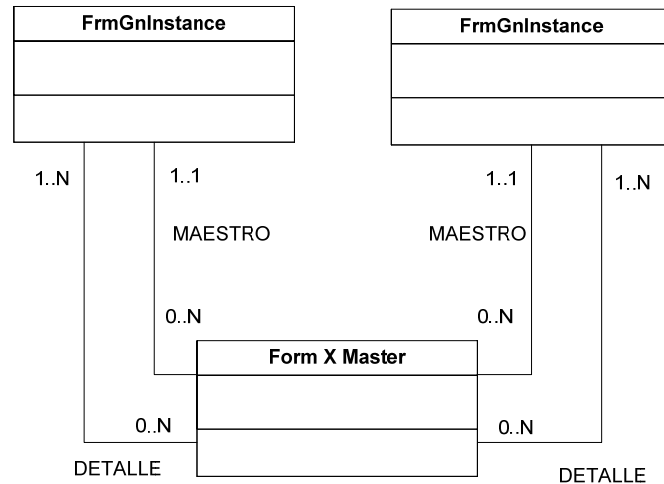


Figura 3.12 Clases generadas a partir de una UI de Maestro/Detalle

Además de las clases que se generan directamente a partir de Unidades de Interacción, están las clases que se generan para ser ejecutadas en la parte servidora. La comunicación entre las clases de la parte cliente y la parte servidora se realiza mediante la clase *Service wrapper*. El objetivo de esta clase es enviar y recibir información al servidor para que se pueda ejecutar la lógica de negocio. Las clases de la parte servidora que se utilizan en la tesis son:

- **Class X action:** Habrá una de estas clases por cada una de las clases definidas en el Modelo de Objetos. La letra X representa cada una de las clases definidas en el Modelo de Objetos. *Class X action* implementa la lógica de negocio de los servicios de cada una de las clases que forman el Modelo de Objetos. Para ello, incluye un método por cada uno de los servicios definidos en la clase del Modelo de Objetos.

- **Class X query:** Es la clase que representa las consultas que se hacen a la base de datos. Al igual que en el caso anterior, habrá una clase de estas por cada una de las clases definidas en el Modelo de Objetos, por eso se utiliza la palabra X en su nombre.

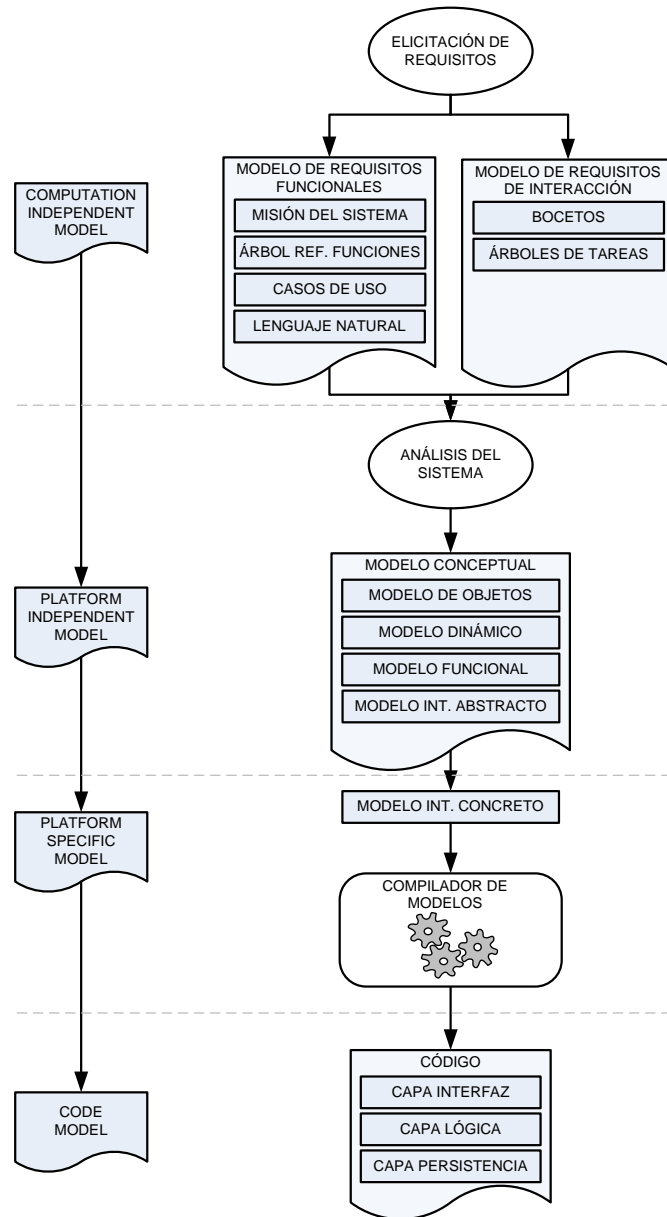


Figura 3.13 Proceso de generación de código de OO-Method

A modo de resumen, la Figura 3.13 hace una comparativa entre los niveles de abstracción de MDA y los modelos de OO-Method. Esta comparativa muestra dónde va ubicado cada modelo de OO-Method dentro del proceso MDA.

La elección de OO-Method como método MDD en el cual incorporar mecanismos de usabilidad, ha estado motivada por dos razones principalmente:

- El modelo conceptual de OO-Method es lo suficientemente abstracto como para incluir primitivas conceptuales nuevas capaces de representar la usabilidad.
- La existencia de la herramienta OlivaNOVA hace posible que la generación de código en base a los modelos conceptuales se haga rápidamente gracias al compilador de modelos y sin necesidad de escribir código manualmente. Esto facilita la evaluación con usuarios de las aplicaciones generadas.

3.4.2.5 Ámbito de las aplicaciones generadas por OO-Method

Las aplicaciones generadas por OO-Method son aplicaciones de gestión, es decir, sistemas de información pensados para empresas. Los entornos que soporta OO-Method son tanto Web como Escritorio, pero no está pensado para dispositivos móviles. En cuanto a la arquitectura de los sistemas, es Cliente/Servidor y no da soporte a sistemas distribuidos.

Las tres principales funcionalidades de los sistemas generados con OO-Method son las siguientes:

- **Introducción de datos mediante formularios:** El usuario puede dar de alta nueva información en el repositorio de información, modificar la ya existente y borrarla en base a formularios.
- **Listado de información:** El usuario puede listar la información almacenada en el repositorio.

- **Ejecutar acciones:** El usuario puede lanzar acciones que modifiquen los datos almacenados conforme a la lógica de negocio del sistema.

En el siguiente capítulo se explica en detalle el método propuesto para incorporar mecanismos de usabilidad en modelos conceptuales OO-Method.

PARTE II: RESOLUCIÓN

Capítulo 4

Método de Incorporación de Mecanismos de Usabilidad a una TTM: MIMAT

Este capítulo muestra en detalle el método propuesto para introducir mecanismos de usabilidad en una TTM. Este método se ha bautizado con el nombre de *MIMAT (Método de Incorporación de Mecanismos de usAbilidad a una TTM)*. Es importante diferenciar entre la figura del diseñador de la TTM y el analista que trabaja con la TTM para la construcción de sistemas. El método MIMAT lo debe aplicar el diseñador a una TTM específica, de manera que una vez incorporados los mecanismos de usabilidad en dicha TTM, el analista pueda utilizarlos en el modelado de nuevos sistemas.

La Figura 4.1 presenta de forma breve en qué consiste el método MIMAT. La idea es partir de los mecanismos de usabilidad descritos en la literatura [Juristo, 2007, a] e incorporarlos en el modelo conceptual de cualquier TTM. Para ello se deben definir primitivas conceptuales que representen los mecanismos de usabilidad en dichos modelos. Las nuevas primitivas conceptuales deben tener la expresividad necesaria para representar todas las cualidades de los mecanismos de usabilidad. Estas primitivas conceptuales pueden pertenecer a cualquiera de las fases tempranas de modelado de la TTM. Utilizando estas primitivas conceptuales, el compilador de modelos debe ser capaz de generar el código que implemente la funcionalidad de los mecanismos de usabilidad. El código generado debe reflejar las opciones de configuración que el analista ha modelado mediante las primitivas conceptuales. Para que esto sea posible es necesario modificar el compilador de modelos con el objetivo de que pueda reconocer

y generar el código para las nuevas primitivas conceptuales. Resumiendo, los cambios que se deben llevar a cabo en una TTM para incorporar mecanismos de usabilidad están centrados en sus primitivas conceptuales y en su compilador de modelos.

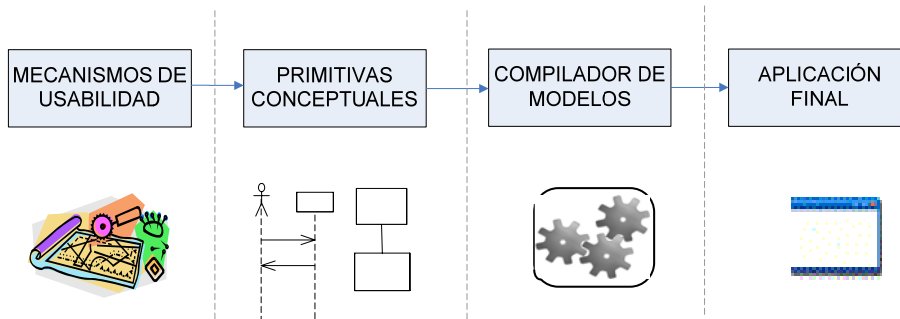


Figura 4.1 Estructura de MIMAT

MIMAT está basado en cuatro pasos, tal y como muestra la Figura 4.2. En dicha figura, los elementos que están en fondo gris se corresponden a los trabajos previos realizados por Juristo. En cambio, los elementos con fondo blanco son la aportación de la tesis. La numeración de la figura coincide con los cuatro pasos que componen el método:

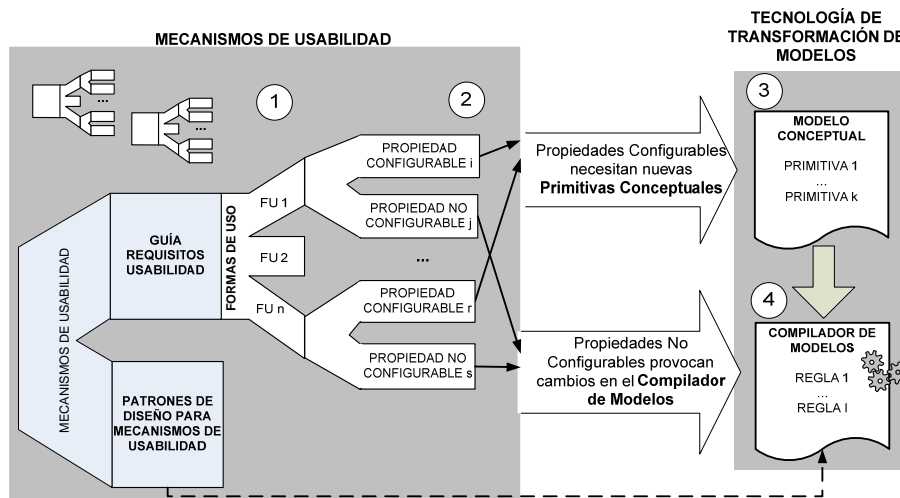


Figura 4.2 Resumen gráfico del método MIMAT

1. Definición de formas de uso para los distintos mecanismos de usabilidad.
2. Identificación de propiedades configurables y propiedades no configurables para cada forma de uso.
3. Definición de primitivas conceptuales para las propiedades configurables.
4. Descripción de los cambios en el compilador de modelos para soportar las propiedades configurables y las no configurables.

El primero de los pasos consiste en estudiar las guías de requisitos de los mecanismos de usabilidad para detectar todas las aplicaciones que puede tener un mecanismo de usabilidad sobre un sistema. Cada una de las distintas aplicaciones se ha denominado **forma de uso**. Un mismo mecanismo de usabilidad puede tener varias formas de uso, tal y como muestra la Figura 4.2 en el paso 1.

Cada forma de uso tiene un conjunto de características para adaptarse a las exigencias de cada usuario. Estas características se conocen con el nombre de **propiedades** dentro del método que proponemos. Estas propiedades se pueden dividir en dos grupos, tal y como se representa en el paso 2 de la Figura 4.2:

- **Propiedades configurables:** Si es necesario que el analista tome alguna decisión para configurar la funcionalidad de esta propiedad.
- **Propiedades no configurables:** Si no es necesario que el analista tome ninguna decisión para configurar este tipo de propiedades.

Una vez se han definido las distintas formas de uso y se han identificado sus propiedades configurables y no configurables, el siguiente paso es identificar los cambios que se deben realizar en la TTM para incorporar las propiedades. La definición de las formas de uso y la identificación de propiedades son genéricas para cualquier TTM, en cambio, los siguientes pasos para definir las primitivas conceptuales y los cambios en el compilador de modelos son dependientes de la TTM elegida para incorporar los mecanismos de usabilidad. Los cambios en el modelo conceptual de la

TTM se realizan con el fin de incorporar primitivas conceptuales que modelen las propiedades configurables de forma abstracta, tal y como se muestra en el paso 3 de la Figura 4.2. Así, mediante estas primitivas el analista puede decidir sobre aquellos aspectos de los mecanismos de usabilidad que sean configurables.

Por último, los cambios en el compilador de modelos van guiados por los patrones arquitectónicos de los mecanismos de usabilidad. Estos cambios se hacen con el objetivo de que el compilador de modelos sea capaz de generar el código que representa las opciones modeladas mediante las nuevas primitivas conceptuales y el código para las propiedades no configurables, tal y como se representa en el paso 4 de la Figura 4.2. El código que se genera de las propiedades no configurables se basa en heurísticos y guías de usabilidad que se aplican de la misma forma para todos los sistemas, porque es un código que no depende de las decisiones del analista.

A continuación se discute cada uno de los cuatro pasos que componen el método MIMAT en una sección.

4.1 Definición de formas de uso

El propósito que se pretende conseguir con un determinado mecanismo de usabilidad se puede alcanzar mediante distintas aplicaciones. Cada una de estas aplicaciones es lo que se ha denominado **forma de uso**. Cada forma de uso es una manera de aplicar el mecanismo de usabilidad en un sistema. Tal y como se ha comentado en capítulos anteriores, los mecanismos de usabilidad disponen de unas guías para especificar los requisitos relacionados con su funcionalidad [Juristo, 2007, b]. Estas guías se definieron con el objetivo de determinar con el usuario cómo aplicar los mecanismos de usabilidad a un determinado sistema y para ello las guías contienen un conjunto de preguntas destinadas al usuario. Estas preguntas tratan de determinar cuál de entre todas las posibles aplicaciones del mecanismo de usabilidad, desea incorporar el usuario al sistema. En esta tesis se han utilizado las preguntas de estas guías para identificar las formas de uso de cada mecanismo de usabilidad.

Para explicar cómo se ha definido cada una de las formas de uso, se ha incluido en su definición las preguntas de la guía de captura de requisitos de la cual se deriva. Además, en la definición de la forma de uso se ha incluido su objetivo. Cada forma de uso tiene un objetivo específico con el que conseguir el propósito del mecanismo de usabilidad. Es decir, cada forma de uso tiene un objetivo concreto con el que satisfacer el propósito más general del mecanismo de usabilidad.

Los objetivos de las formas de uso de un mismo mecanismo de usabilidad no se contradicen entre sí, sino que intentan conseguir el propósito del mecanismo mediante distintas aplicaciones. Por lo tanto, las formas de uso no son excluyentes entre sí, es decir, un mismo sistema puede incorporar distintas formas de uso de un mismo mecanismo de usabilidad. Al ser el propósito de todas las formas de uso el mismo, su incorporación al sistema hace que se complementen entre sí y ayuden a mejorar la usabilidad del sistema.

Las formas de uso representan cada una de las aplicaciones de un mecanismo de usabilidad, pero es necesaria más información para que el analista pueda satisfacer los requisitos del usuario. Las formas de uso pueden tener varias alternativas en su configuración. A estas alternativas de configuración se les ha denominado propiedades. En la siguiente sección se detallará el significado de estas propiedades y sus tipos.

4.2 Identificación de propiedades

Las guías de los mecanismos de usabilidad contienen más información además de las distintas aplicaciones que puede tener un mecanismo de usabilidad (formas de uso). Estas guías también incluyen preguntas para determinar aspectos sobre cómo configurar las distintas aplicaciones del mecanismo de usabilidad que no están representadas mediante las formas de uso. Por lo tanto, es necesario un nuevo elemento para representar las distintas opciones de configuración de cada una de las formas de uso. Estos elementos se han denominado propiedades.

Las **propiedades** de las formas de uso son las distintas posibilidades de configuración que tiene la forma de uso para adaptarse a los requisitos de

usabilidad del usuario. En la mayoría de los casos, estas propiedades las debe adaptar el analista a un sistema en particular. En otros casos, algunas de estas propiedades las puede configurar el compilador de modelos de la TTM de forma automática sin la intervención del analista. Por tanto, se distinguen dos tipos de propiedades: configurables y no configurables.

4.2.1 Propiedades configurables

Este tipo de propiedades está formado por aquellas que requieren que el analista decida ciertos aspectos de su configuración. La configuración óptima de estas propiedades depende de los requisitos impuestos por el usuario para un caso particular y por tanto, es el analista el que debe especificarlas en base a estos requisitos.

Las distintas alternativas de configuración de estas propiedades configurables se han extraído de las guías donde se capturan los requisitos para los mecanismos de usabilidad [Juristo, 2007, b] y de ciertas guías de usabilidad como los patrones de usabilidad de Tidwell [Tidwell, 2005], Welie [Welie, 2000], Coram [Coram, 1996], Benson [2002] y Brighton [Griffiths, 2002].

4.2.2 Propiedades no configurables

Al igual que hay propiedades cuya configuración puede variar dependiendo de los requisitos del usuario, hay otras propiedades que no presentan alternativas a su configuración. Son propiedades que, o bien no presentan alternativa en las guías del mecanismo de usabilidad al que pertenezcan, o bien las guías recomiendan que su configuración sea siempre de la misma manera en todos los sistemas desarrollados. Este tipo de propiedades no pueden ser configuradas por el analista porque su configuración debe ser la misma para todos los sistemas desarrollados.

Dentro de una TTM, la configuración de estas propiedades se puede delegar al compilador de modelos. El compilador de modelos será el encargado de incorporar las propiedades no configurables en todos los sistemas, garantizando que se aplicará la misma configuración en todos ellos. De esta manera se mejora la eficiencia en el desarrollo de software,

ya que la incorporación de las propiedades no configurables se hace de forma totalmente automática.

Tanto las formas de uso como sus propiedades son válidas para cualquier TTM. En cambio, los otros dos pasos que restan del método propuesto para incorporar los mecanismos de usabilidad a una TTM (definición de primitivas conceptuales y cambios necesarios en el compilador de modelos) son dependientes de la TTM. A continuación se describen ambos pasos.

4.3 Definición de primitivas conceptuales

El primero de los cambios que debe incorporar el diseñador a la TTM es a nivel conceptual. Este paso es por tanto dependiente de la TTM seleccionada para incorporar los mecanismos de usabilidad, ya que los modelos conceptuales de cada TTM son exclusivos. Es decir, hay tantos modelos conceptuales como TTMs. En este paso del método, el modelo conceptual de la TTM elegida para añadir los mecanismos de usabilidad se debe enriquecer con primitivas conceptuales que tengan la expresividad suficiente para representar las propiedades de las formas de uso de manera abstracta.

De los dos tipos de propiedades que se han definido, sólo las propiedades configurables implican cambios a nivel conceptual. Para incorporar las propiedades configurables a la TTM es necesario incluir nuevas primitivas conceptuales que representen las posibles configuraciones de estas propiedades. El analista, a través de estas nuevas primitivas conceptuales, debe ser capaz de modelar las propiedades configurables según los requisitos del usuario.

En este paso de MIMAT, el diseñador debe verificar si para cada una de las propiedades configurables, existe ya alguna primitiva conceptual que la represente en el actual modelo conceptual de la TTM. En caso de que ya exista representación conceptual para que el analista pueda configurar esa propiedad con todas sus alternativas, no es necesario introducir nuevas primitivas para representarla. En caso de no existir forma de representar la

propiedad configurable o alguna de las posibilidades de configuración de una propiedad no se pueda representar, el diseñador debe añadir la expresividad correspondiente al modelo conceptual de la TTM. Los pasos para enriquecer el modelo conceptual para incorporar las propiedades configurables son:

1. **Identificar en qué modelo (vista del modelo conceptual)** de los que forman el modelo conceptual se debe representar cada una de las propiedades configurables del mecanismo de usabilidad. Esta selección depende de qué propiedad del mecanismo de usabilidad se desea configurar:
 - Si la propiedad afecta a aspectos visuales, el modelado de ésta se llevará a cabo en el modelo donde se represente la interfaz de forma abstracta.
 - Si la propiedad afecta a la persistencia de datos, ésta se modelará en el modelo que representa la persistencia de información en el sistema.
 - Si la propiedad afecta a la funcionalidad del sistema, el modelado de ésta se representará en el modelo donde se represente la funcionalidad del sistema.

Dependiendo de la TTM, puede que haya más o menos modelos que se puedan ver afectados por la incorporación de las propiedades.

2. **Identificar las primitivas** a incluir dentro del modelo. Cada una de las propiedades configurables necesita de al menos una primitiva conceptual para representarse de forma abstracta dentro de la TTM. En este paso, el diseñador identifica las primitivas necesarias para representar la propiedad dentro del modelo seleccionado en el punto uno. Las primitivas identificadas deben tener la capacidad de representar todas las alternativas de configuración de la propiedad.

Además de las configurables, también hay propiedades no configurables. Las propiedades no configurables no implican ningún cambio en el modelo conceptual de la TTM. La incorporación al sistema de estas propiedades la

llevará a cabo el compilador de modelos, tal y como se explica en la siguiente sección.

4.4 Cambios necesarios en el compilador de modelos

El último paso de MIMAT consiste en decidir los cambios que se deben aplicar al compilador de modelos, es decir, a la estrategia de generación de código. Este paso, al igual que el de la definición de primitivas conceptuales, es dependiente de la TTM seleccionada porque cada compilador de modelos genera el código del sistema de forma exclusiva. Los cambios a incorporar en el compilador de modelos de la TTM seleccionada, tal y como muestra la Figura 4.2, se derivan de:

- **Las nuevas primitivas conceptuales que representan las propiedades configurables:** El compilador de modelos debe ser capaz de reconocer estas nuevas primitivas conceptuales y generar el código que las implemente con la misma configuración que haya representado el analista en el modelo conceptual. Para ello, el compilador de modelos debe ser capaz de generar el código que implemente toda la expresividad representada mediante las primitivas conceptuales. Es decir, debe ser capaz de implementar todas las posibles alternativas de configuración que se pueden representar con las propiedades configurables.
- **Las propiedades no configurables:** Estas propiedades, aunque no implican cambios a nivel conceptual, sí afectan al compilador de modelos. Éste debe generar la funcionalidad de las propiedades no configurables de forma automática sin la intervención del analista.

Los dos tipos de cambios implican que se añada al código generado nuevos atributos, métodos y clases con respecto a los que ya genere actualmente el compilador de modelos. En trabajos de Juristo [Juristo, 2007, a] se propone una solución en base a patrones de diseño para implementar los mecanismos de usabilidad. Esta solución describe cómo incorporar los mecanismos de usabilidad a la arquitectura de los sistemas. La descripción

puede servir como punto de partida sobre cómo implementar las nuevas primitivas conceptuales y las propiedades no configurables. Por lo tanto, los cambios a incorporar en el compilador de modelos serán guiados por los patrones de diseño propuestos para implementar los mecanismos de usabilidad.

Para describir los cambios sobre el compilador de modelos se ha utilizado el concepto de código genérico. Se ha definido el **código genérico** como el código en un lenguaje de programación que representa de forma abstracta cualquier sistema generado mediante una TTM concreta. El código genérico tiene en cuenta sólo las clases que se generan, sus atributos y sus métodos en forma genérica, pero sin prestar atención a los detalles de la lógica de negocio, ya que lo expresado mediante este código debe ser válido para cualquier sistema. Cada sistema desarrollado mediante la TTM es una instancia de este código genérico. El código genérico se va a representar mediante dos diagramas UML [UML, 2008]:

- **Diagramas de Clase:** Se utilizan para representar las clases que son necesarias incorporar al código para implementar los mecanismos de usabilidad, la relación de estas nuevas clases con las que ya se generan actualmente, y los atributos y métodos que representan la funcionalidad de los mecanismos de usabilidad. Este diagrama da una visión global de los cambios que introduce un mecanismo de usabilidad específico.
- **Diagramas de Secuencia:** El Diagrama de Clases sólo representa las clases, atributos y métodos necesarios para representar los mecanismos de usabilidad, pero no representa la forma en la que las clases interactúan entre sí mediante los métodos. Para representar este aspecto es necesario utilizar los Diagramas de Secuencia. Estos diagramas se utilizan para expresar la secuencia de invocación de métodos entre las clases que representan los mecanismos de usabilidad. Las clases que se muestran en este diagrama deben ser las mismas que las descritas mediante el Diagrama de Clases.

Tal y como muestra la Figura 4.3, cada uno de los mecanismos de usabilidad se representa mediante un Diagrama de Clases, mientras que

cada una de las formas de uso que tenga el mecanismo de usabilidad se representa mediante un Diagrama de Secuencia. Habrá tantos Diagramas de Secuencia como formas de uso porque cada aplicación de un mecanismo de usabilidad requiere unos métodos distintos para su implementación y por lo tanto, una secuencia de invocación entre métodos distintos. La construcción de los Diagramas de Clase y los de Secuencia viene guiada por los patrones de diseño propuestos por Juristo.

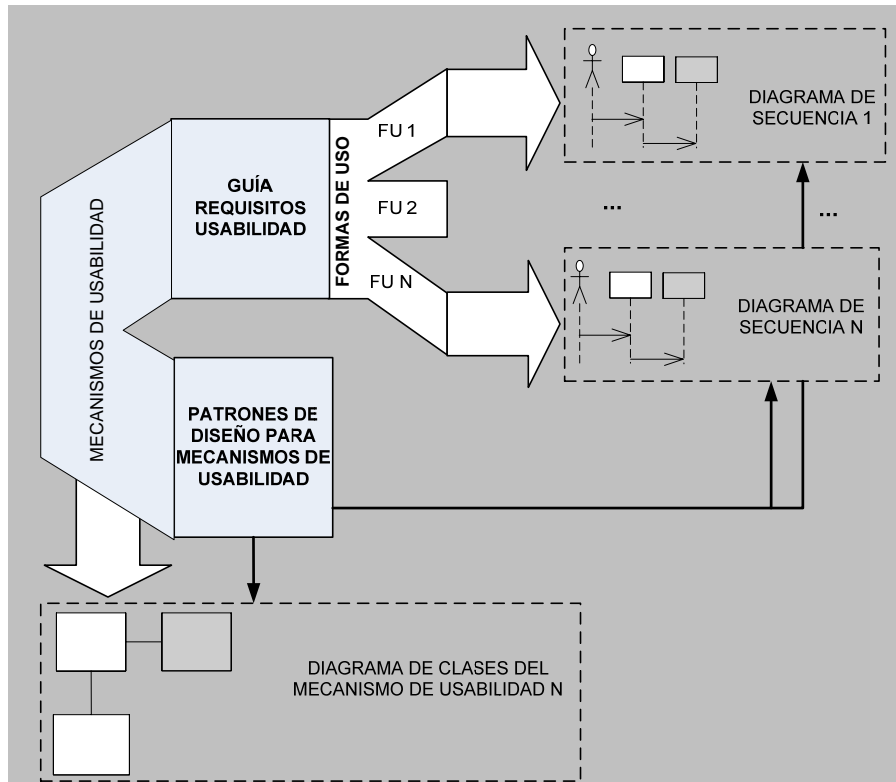


Figura 4.3 Representación de los mecanismos de usabilidad mediante Diagramas de Clase y de Secuencia

Ambos diagramas incluyen tres tipos de clases representadas con una notación gráfica distintiva para diferenciarlas entre sí (Figura 4.4):

- **Clases nuevas:** Son aquellas clases que no existen actualmente en el compilador de modelos. Se representan con un color gris.

- **Clases modificadas:** Son clases que ya existen en el compilador de modelos pero que se ven modificadas por la incorporación de los mecanismos de usabilidad, bien añadiendo o modificando atributos y métodos. Se representan con un trazo rallado.
- **Clases que no se ven alteradas:** Son clases que ya existen en el compilador de modelos y no se ven modificadas por la incorporación de los mecanismos de usabilidad, pero que son necesarias en los diagramas para entender el funcionamiento de los mecanismos de usabilidad. Se representan con color blanco.

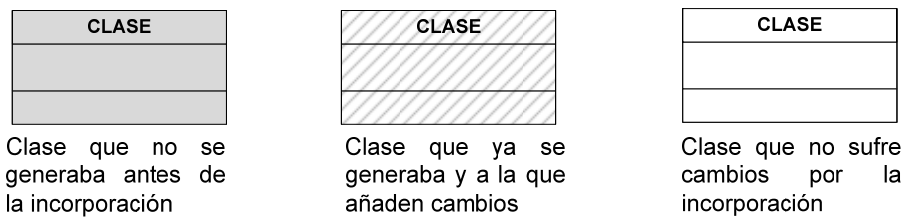


Figura 4.4 Tipos de representación de las clases al incorporar los mecanismos de usabilidad

4.5 Incorporación de mecanismos de usabilidad a OO-Method

En los siguientes capítulos se muestra la aplicación del método MIMAT a una TTM concreta: OO-Method. La tesis contiene un capítulo por cada uno de los FUFs definidos por Juristo [Juristo, 2007, b]. En cada capítulo se detalla para cada mecanismo de usabilidad en los que se divide el FUF: las formas de uso; las propiedades; las nuevas primitivas conceptuales necesarias para su representación abstracta; y los cambios en el compilador de modelos para implementar el código que soporte su funcionalidad.

La definición de formas de uso y propiedades es aplicable a cualquier TTM, ya que son conceptos totalmente independientes de la estrategia de generación de código. En cambio, la definición de nuevas primitivas y los cambios en el compilador de modelos son específicos para OO-Method

porque están basados en su estrategia de modelado conceptual y de generación de código. Por ejemplo, el Modelo de Interacción Abstracto contiene una serie de primitivas que son específicas para OO-Method, ya que otras TTMs no contienen ningún modelo para representar la interacción o bien son primitivas, en general, diferentes a las de OO-Method. Las nuevas primitivas derivadas de las formas de uso deben estar en sintonía con las primitivas de OO-Method ya existentes, y por tanto, son dependientes de ellas. Además, la estrategia de generación de código depende de las primitivas conceptuales, y éstas a su vez dependen de la TTM. Por lo tanto, por transitividad, se puede afirmar que los cambios en el compilador de modelos son específicos de una TTM. Aun así, lo definido para OO-Method sirve de guía a la hora de aplicar MIMAT a cualquier otra TTM. Al aplicar MIMAT sobre distintas TTMs, cambiarán detalles sobre la forma de representar las nuevas primitivas conceptuales y en la estrategia de generación de su código, pero la semántica de las modificaciones derivadas de MIMAT será la misma.

En la definición de las formas de uso y las propiedades se incluyen las preguntas de la guía de captura de requisitos a partir de la cuales se han derivado. En todos los casos estudiados, la guía de la que se derivan las formas de uso y las propiedades de un mecanismo de usabilidad es siempre la guía que captura los requisitos del propio mecanismo de usabilidad.

Algunas propiedades no proceden de las guías, sino de heurísticos, como los de Nielsen [Nielsen, 1993] o de patrones de usabilidad descritos en la literatura, como los de Tidwell [Tidwell, 2005]. En estos casos se justifica de manera precisa la procedencia de dichas propiedades.

Algunas de las propiedades ya están soportadas actualmente por OO-Method, por lo tanto antes de proponer nuevas Primitivas Conceptuales hay que detectar las propiedades que ya están soportadas por OO-Method. Una vez detectadas las propiedades que no están aun soportadas, el siguiente paso es definir nuevas Primitivas Conceptuales con las que representarlas de forma abstracta.

Es importante remarcar que, aunque en el paso dos de MIMAT se clasifican las propiedades en configurables y no configurables, a la hora de aplicar las propiedades a OO-Method, algunas propiedades configurables se pueden

convertir en no configurables y viceversa. Esto se debe a que al aplicar las propiedades a un contexto específico, algunos de los tipos de estas propiedades no tienen sentido. Las propiedades que sufren este cambio las denominamos **propiedades establecidas**, porque el tipo de estas propiedades (configurable o no configurable) viene establecido por la TTM y no por las preguntas de las guías de captura de requisitos. Es decir, por el tipo de aplicaciones que genera OO-Method, algunas propiedades configurables no tiene sentido que sean configurables y viceversa. En los siguientes capítulos se verán ejemplos de este tipo de propiedades.

Para cada una de las propiedades configurables que no estén soportadas aun por OO-Method, se proponen primitivas conceptuales para representarlas de manera abstracta. Las nuevas primitivas conceptuales se presentan agrupadas por la vista del modelo conceptual que modifican. Una misma forma de uso puede afectar a más de una vista, dependiendo de las capas a las que afecte su incorporación: persistencia, funcionalidad o interfaz.

Finalmente, se describen los cambios en el compilador de modelos con el objetivo de soportar las nuevas primitivas conceptuales y las propiedades no configurables. Esta descripción está basada en las clases del código genérico que el compilador de modelos genera para C#, concretamente para la versión de C# 2003. Para ayudar en la descripción, se han utilizado Diagramas de Clase y de Secuencia.

En la siguiente sección se presenta un caso de ilustración que será utilizado en los siguientes capítulos para ilustrar la aplicación de MIMAT a OO-Method.

4.6 Caso de ilustración

El caso de ilustración elegido para ilustrar la aplicación de MIMAT es un sistema para realizar el alquiler de coches. Se ha elegido este sistema porque es un ejemplo representativo del tipo de aplicaciones que genera OO-Method: aplicaciones de gestión. Este es un caso de ilustración ficticio, es decir, las capturas de pantalla utilizadas no son de un sistema funcional. Este sistema está creado con una arquitectura cliente/servidor, donde varios

clientes acceden a una misma base de datos ubicada en el servidor. Los usuarios de este sistema son trabajadores de la empresa de alquiler de coches que interactúan con el sistema para atender las peticiones de alquiler de los clientes. Las operaciones más comunes de este sistema son:

- **Reservar alquiler de coche:** El usuario introduce los datos del cliente que va a reservar el alquiler, las fechas, la información bancaria y el tipo de coche que va a alquilar. A partir de estos datos, se realizará el alquiler.
- **Recoger coche:** El usuario retira el coche que tenía reservado previamente
- **Devolver coche:** El usuario devuelve el coche que había recogido previamente.
- **Alta de cliente:** El usuario introduce los datos personales de un cliente nuevo y se almacenan en la base de datos.
- **Alta de coche:** Cada vez que la empresa adquiere un nuevo coche se introducen sus características y se almacenan en la base de datos.
- **Baja de coche:** Cuando los coches se retiran de la empresa por viejos o averiados pasan al estado de retirados y dejan de estar disponibles para el alquiler.
- **Imprimir factura:** El cliente puede solicitar que se le imprima la factura de algunos de los alquileres que ya haya realizado.
- **Listar coches en alquiler:** Muestra un listado con todos los coches disponibles.
- **Poner coche en venta:** Mediante este servicio se selecciona un coche para ponerlo en venta. Al ponerlo en venta deja de estar disponible para alquilar.
- **Listar coches en venta:** Muestra un listado con todos los coches en venta.

Estas operaciones del sistema son las que se utilizan para mostrar las aplicaciones de las formas de uso. El Diagrama de Clases que representa el sistema es el mostrado en la Figura 4.5.

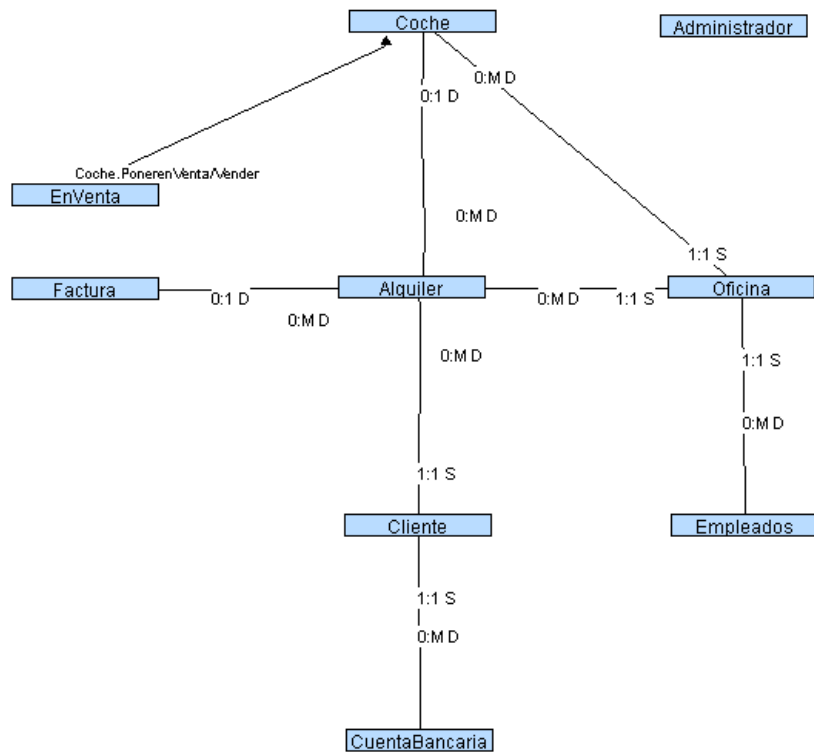


Figura 4.5 Diagrama de Clases del caso de ilustración

A continuación, se detallan los atributos y los servicios de cada una de las clases:

- La clase *Cliente* tiene como atributos: el nombre del cliente, los apellidos, el DNI, la fecha de nacimiento, el domicilio, el número de portal, el número de puerta, el código postal, si posee carné de conducir y su estado civil (soltero o casado). Los servicios de esta clase son: alta de cliente, modificar cliente y borrar cliente.
- La clase *CuentaBancaria* representa los datos bancarios del cliente. Los atributos de esta clase son: el número de cuenta, el saldo, el nombre del banco, la fecha de caducidad y el IBAN. Los servicios de esta clase son: crear cuenta, modificar cuenta y borrar cuenta.

- La clase *Alquiler* tiene como atributos: la fecha en la que está previsto recoger el coche, la fecha en la que está previsto devolver el coche, la fecha en la que realmente se recoge el coche y la fecha en la que realmente se devuelve el coche. Los servicios de esta clase son: reservar alquiler de coche, borrar reserva, modificar reserva, recoger coche, devolver coche.
- La clase *Factura* tiene como atributos: la fecha de emisión y si está pagada o no. Los servicios de esta clase son: crear factura, modificar factura, borrar factura, pagar factura, imprimir factura.
- La clase *Coche* tiene como atributos: la marca, el modelo, la matrícula, el número de puertas, el tipo de combustible, el color, los caballos, si dispone de radio CD y si dispone de aire acondicionado. Los servicios de esta clase son: crear coche, modificar coche, borrar coche, poner coche en venta.
- La clase *EnVenta* representa los coches que han pasado de estar en alquiler a estar en venta. Tiene como atributos los heredados por coche además de: la fecha en la que se puso en venta, el precio. Esta clase tiene sólo el servicio vender.
- La clase *Oficina* representa las distintas sucursales desde las que se pueden alquilar coches. Tiene como atributos: el nombre, la provincia, la localidad, la dirección, el número, el código postal y el número de teléfono. Los servicios de esta clase son: crear oficina, modificar oficina y borrar oficina.
- La clase *Empleados* representa los trabajadores que hay en cada oficina. Los atributos de esta clase son: el nombre, los apellidos y el DNI. Los servicios son: crear empleado, modificar empleado y borrar empleado.
- La clase *Administrador* es la clase agente que puede ejecutar cualquier servicio ofrecido por el sistema.

Este caso de ilustración se utilizará a lo largo de los siguientes capítulos de la tesis para ilustrar la aplicación de MIMAT.

Capítulo 5

Incorporación de Feedback

Feedback es un FUF que tiene como objetivo mantener informado al usuario sobre lo que está ocurriendo en todo momento dentro del sistema. A continuación se presenta una sección por cada mecanismo de usabilidad en los que se divide: *System Status Feedback*, *Interaction Feedback*; *Progress Feedback*; *Warning*. En cada sección se aplica el método MIMAT a un mecanismo de usabilidad distinto.

5.1 System Status Feedback

El objetivo principal que se pretende alcanzar con la incorporación de este mecanismo de usabilidad es el de informar al usuario de cambios relevantes que se producen en el sistema o cuando se produce un fallo en los siguientes contextos: durante la ejecución de un servicio; antes de la ejecución de un servicio; después de la ejecución de un servicio, porque no hay suficientes recursos; porque los recursos externos no están trabajando adecuadamente. Este mecanismo de usabilidad comparte el mismo objetivo que el heurístico de Nielsen *Visibility of system status* [Nielsen, 1993].

5.1.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *System Status Feedback* (Anexo I) se han obtenido las siguientes formas de uso:

1. WU_SSF1: Informar del éxito o fracaso en la ejecución del servicio.
2. WU_SSF2: Mostrar el estado de la información.

3. WU_SSF3: Mostrar el estado de las acciones
4. WU_SSF4: Informar de falta de recursos
5. WU_SSF5: Informar de fallos en dispositivos externos

La Tabla 5.1 muestra una visión global de las formas de uso, las preguntas de la guía de captura de requisitos a partir de las cuales se han derivado y los objetivos que pretenden conseguir cada una de ellas.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Quiere el usuario que el sistema le informe sobre sus fallos?	Informar de si la ejecución de un servicio ha terminado en error o en éxito	Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)
¿Tiene el sistema la capacidad de anunciar su estado actual?	Informar del estado de la información almacenada en el sistema antes de la ejecución de un servicio	Mostrar el estado de la información (WU_SSF2)
	Evitar errores mostrando el estado de las acciones que puede lanzar el usuario	Mostrar el estado de las acciones (WU_SSF3)
¿Quiere el usuario que el sistema le notifique cuando no hay suficientes recursos en el sistema para ejecutar un servicio?	Informar de la falta de recursos del sistema para ejecutar un servicio	Informar de falta de recursos (WU_SSF4)
	Informar de un fallo en un recurso externo al sistema que impide la ejecución de servicios	Informar de fallos en dispositivos externos (WU_SSF5)

Tabla 5.1 Derivación de las formas de uso de *System Status Feedback* a partir de la guía de captura de requisitos

La primera de las formas de uso, **Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)** se deriva de la pregunta de la guía de captura de requisitos *¿Quiere el usuario que el sistema le informe sobre sus fallos?* De los posibles fallos que puede tener un sistema software, esta forma de uso se centra en los fallos que se producen por la petición de ejecución de un servicio, ya que es un fallo típico en sistemas software de gestión. Por lo tanto, la forma de uso que representa la aplicación de este requisito tiene como objetivo el de informar si la ejecución de un servicio finaliza en fallo o en éxito. De esta manera se garantiza que el usuario sea consciente del estado del sistema tras la solicitud de ejecución del servicio.

Por ejemplo, para el sistema de alquiler de coches, cada acción que solicite el usuario (reservar alquiler de coche, devolver coche, dar de alta cliente, etc.) debe ir acompañada de un mensaje que muestre si la acción se ha ejecutado o no con éxito.

La segunda forma de uso, **Mostrar el estado de la información (WU_SSF2)**, se deriva de la pregunta de la guía de captura de requisitos *¿Tiene el sistema la capacidad de anunciar su estado actual?* El objetivo de esta forma de uso es el de mostrar el estado del sistema antes de que el usuario solicite ejecutar un servicio. Es decir, muestra el estado del sistema dependiendo de la información almacenada (información histórica), sin tener en cuenta el siguiente servicio que vaya a ejecutar el usuario. Por ejemplo, en el sistema de alquiler de coches es imprescindible que antes de permitir al usuario hacer la reserva de un coche se muestre el número y modelo de coches disponibles.

La forma de uso **Mostrar el estado de las acciones (WU_SSF3)** se deriva de la pregunta de la guía, *¿Tiene el sistema la capacidad de anunciar su estado actual?* El objetivo de esta forma de uso es el de indicar qué acciones no se pueden ejecutar en un estado concreto del sistema. Es decir, esta forma de uso evita que surjan errores porque el usuario ejecute una acción en un estado en el cual no es posible. Para prevenir este tipo de errores, la aplicación de WU_SSF3 muestra qué acciones están habilitadas en el estado actual del sistema y cuáles no están habilitadas.

Aunque la procedencia de WU_SSF2 y WU_SSF3 sea la misma pregunta de la guía de captura de requisitos, en esta tesis se diferencian entre sí en

base a su finalidad. Mientras que en WU_SSF2 la finalidad de mostrar el estado del sistema es simplemente informativa, en WU_SSF3 la finalidad es la de prevenir errores en el usuario. La forma de uso WU_SSF3 es equivalente al patrón de Tidwell llamado *Inhabilitar acciones permitidas* [Tidwell, 2005] y tiene la misma finalidad que el heurístico de Nielsen llamado *Error prevention* [Nielsen, 1993] y que el criterio ergonómico de Bastien y Scapin *Error protection* [Bastien, 1993]. Por ejemplo, en el sistema de alquiler de coches, el botón de imprimir factura sólo se habilitará cuando el cliente haya terminado de hacer la reserva del coche.

Por lo que respecta a la forma de uso **Informar de falta de recursos (WU_SSF4)**, se deriva de la pregunta de la guía de captura de requisitos, *¿Quiere el usuario que el sistema le notifique cuándo no hay suficientes recursos en el sistema para ejecutar un servicio?* El objetivo de esta pregunta es el de informar al usuario qué servicios no se pueden ejecutar por falta de recursos del sistema. Esta forma de uso es necesaria porque hay servicios muy complejos que pueden llegar a necesitar más recursos de los que hay disponibles en un determinado momento en el sistema. Este problema sólo se presentará en aquellos servicios que requieren de una gran cantidad de recursos para su ejecución. Por ejemplo, si al sistema de alquiler de coches acceden simultáneamente muchos usuarios, puede que no haya la memoria suficiente para llevar a cabo el alquiler.

Por último, la forma de uso **Informar de fallos en dispositivos externos (WU_SSF5)** se deriva de la pregunta *¿Quiere el usuario que el sistema le notifique si hay un problema con recursos externos u otros dispositivos que interactúan con el sistema?* El objetivo de esta forma de uso es informar al usuario cuando no se pueda ejecutar un servicio por causa de un recurso externo al sistema. Muchos de los servicios ofertados por un sistema pueden requerir dispositivos externos, como por ejemplo, impresoras o lectores de CD-ROM. En caso de que alguno de estos dispositivos externos a la aplicación falle, el usuario debe ser informado de este fallo. Por ejemplo, en el sistema de alquiler de coches, una vez finalizada la reserva del coche, si falla la impresora y no se puede imprimir la factura, el sistema debe avisar al usuario que se ha producido un fallo en la impresión.

En la siguiente sección se explican las propiedades que componen cada una de estas formas de uso.

5.1.2 Propiedades

A continuación se explican las propiedades que adaptan las formas de uso a los requisitos de usabilidad exigidos por el usuario.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿De qué fallos quiere ser avisado el usuario?	Indicar los servicios en los que el usuario quiere ver si se han ejecutado o no correctamente	Selección de servicios (P1_WU_SSF1)	No configurable
¿Qué información se debe mostrar de forma bloqueante porque es crítica? ¿Qué información se debe resaltar porque es importante pero no crítica? ¿Qué información se debe mostrar en el área de estado?	Determinar el aspecto visual de la información que muestra si el servicio se ha ejecutado o no correctamente	Visualización del mensaje (P2_WU_SSF1)	Configurable

Tabla 5.2 Derivación de propiedades de la forma de uso *Informar del éxito o fracaso en la ejecución del servicio* a partir de la guía de captura de requisitos

Las **propiedades de Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)** determinan qué servicios informarán del éxito o fracaso y cómo se visualizará esta información. La Tabla 5.2 muestra de forma resumida cómo se han derivado las propiedades a partir de las preguntas de la guía de captura de requisitos.

Más detalladamente, las propiedades son las siguientes:

- **Selección de servicios (P1_WU_SSF1):** Esta propiedad se deriva de la pregunta extraída de la guía, *¿De qué fallos quiere ser avisado el usuario?* Una vez que el usuario tiene claro su deseo de que el sistema le informe de los fallos y éxitos en la ejecución de servicios, esta propiedad especifica qué acciones proporcionarán al usuario dicha información. Es decir, de entre todos los servicios del sistema, cuáles van a poder informar al usuario de su éxito o su fracaso. La propiedad de selección de servicios es no configurable, ya que según el criterio ergonómico de Bastien y Scapin llamado *Immediate feedback* [Bastien, 1993], se recomienda que el sistema muestre esta información siempre que sea posible. Por lo tanto, proponemos delegar al compilador de modelos la incorporación de esta propiedad en el sistema.
- **Visualización del mensaje (P2_WU_SSF1):** Las preguntas de la guía de las cuales se extrae esta propiedad son: *¿Qué información se debe mostrar de forma bloqueante porque es crítica? ¿Qué información se debe resaltar porque es importante pero no crítica? ¿Qué información se debe mostrar en el área de estado?* Todas estas preguntas pretenden detectar la manera concreta en la que el usuario quiere visualizar los mensajes de aviso. La retroalimentación se puede representar en el sistema mediante iconos o mediante mensajes textuales que el propio analista debe definir. Cada tipo de visualización contiene a su vez distintas alternativas relacionadas con los posibles iconos a utilizar y con el formato del mensaje de aviso. Además, ambos tipos de visualización se pueden mostrar en una nueva ventana emergente o desde la misma ventana utilizada para lanzar el servicio. Estas decisiones las debe tomar el analista mediante una propiedad configurable para que se le puedan asignar todos los posibles valores de visualización.

Como ejemplo para ilustrar la aplicación de estas propiedades a un sistema real, esta tesis se va a centrar en el servicio alta de cliente del sistema de alquiler de coches. La Figura 5.1 muestra la ventana en la que el usuario debe introducir los valores para darse de alta como cliente.

The screenshot shows a window titled "Alta de cliente" with a blue header. It is divided into three main sections:

- Identificación:** Contains text boxes for "Nombre:" (Luis), "Apellidos:" (Pérez Martínez), "DNI:" (335687902L), and "Fecha de nacimiento:" (7/11/1981).
- Dirección:** Contains text boxes for "Domicilio:" (Colón), "Número:" (AA), "Puerta:" (3), "Provincia:" (Valencia), "Localidad:" (Alboraya), and "Código postal:" (46120).
- Otros datos:** Includes a checked checkbox "Posee Carné de conducir" and a "Estado civil" section with radio buttons for "Soltero" (selected) and "Casado".

At the bottom right, there are two buttons: "Aceptar" and "Cancelar".

Figura 5.1 Alta de cliente que informa sobre el éxito o fracaso de su ejecución

En el caso de ejemplo, la propiedad *Selección de servicios (P1_WU_SSF1)* al ser no configurable y abarcar a todos los servicios definidos, también incluye el servicio alta de cliente. En cuanto a la propiedad *Visualización del mensaje (P2_WU_SSF1)*, el analista ha decidido usar para los mensajes de error una nueva ventana emergente con un texto introducido por él y para los mensajes de éxito mostrar un icono en la ventana principal.

Cuando el usuario pulse el botón *Aceptar* y se inicie la ejecución, el sistema comprobará los datos y se mostrará un mensaje de error porque el número de domicilio (campo *Número*) es una cadena de caracteres y no un valor numérico. Por lo tanto, se mostrará al usuario un error como el de la Figura 5.2.

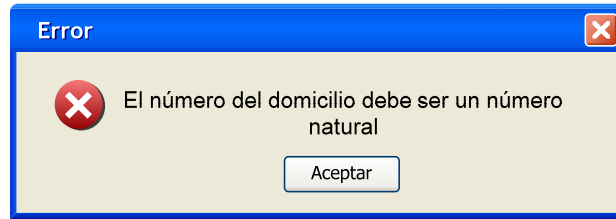


Figura 5.2 Mensaje de error para el servicio alquiler de coche

Cuando el usuario corrija el error, cambie el valor AA por el 70 y vuelva a lanzar otra vez la ejecución del servicio, éste terminará satisfactoriamente. El éxito de la ejecución se mostrará con un icono en la ventana principal, tal y como muestra la Figura 5.3.

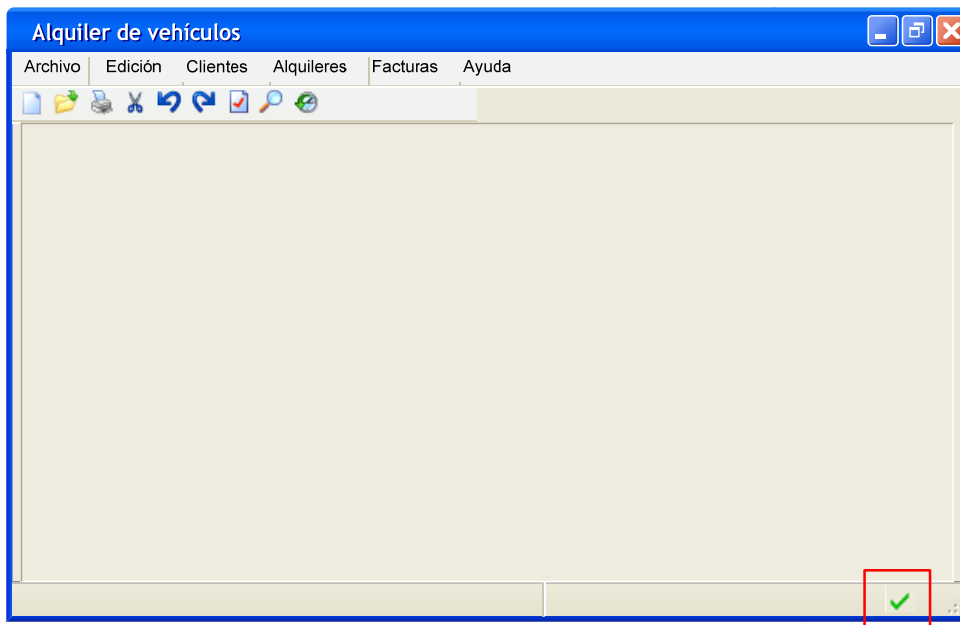


Figura 5.3 Mensaje de éxito para el servicio alquiler de coche

En cuanto a las **propiedades de Mostrar el estado de la información (WU_SSF2)**, determinan qué información almacenada sobre el estado del sistema se va a mostrar y cómo se va a mostrar. La Tabla 5.3 muestra una visión global de las preguntas de la guía de captura de requisitos de las que

se han derivado las propiedades y los objetivos que se pretenden conseguir con cada una de ellas.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué información se debe mostrar al usuario?	Indicar la parte del mensaje que pertenece a datos almacenados en el sistema	Información dinámica a visualizar (P1_WU_SSF2)	Configurable
¿Qué información se debe mostrar al usuario?	Indicar la parte del mensaje que no varía en todos los estados del sistema.	Información estática a visualizar (P2_WU_SSF2)	Configurable
¿Qué información se debe mostrar de forma bloqueante porque es crítica? ¿Qué se debe resaltar porque es importante pero no crítico? ¿Qué se debe mostrar en el área de estado?	Determinar el aspecto visual del mensaje creado a partir de la parte dinámica y la estática	Visualización del mensaje (P3_WU_SSF2)	Configurable

Tabla 5.3 Derivación de propiedades de la forma de uso *Mostrar el estado de la información* a partir de la guía de captura de requisitos

Tal y como se puede apreciar, todas las propiedades son configurables. A continuación se van a explicar cada una de las propiedades de forma más detallada:

- **Información dinámica a visualizar (P1_WU_SSF2):** La pregunta de la guía de la que se deriva esta propiedad es, *¿Qué información se debe mostrar al usuario?* Esta propiedad determina los atributos de los objetos almacenados en la base de datos del sistema que se van a visualizar. En algunos casos, la información a mostrar debe ser más elaborada que si se tratara de un simple atributo, es decir, la información a mostrar puede depender de varios atributos o de fórmulas complejas. Para ello se debe dotar a esta propiedad de la capacidad de definir fórmulas que especifiquen de forma precisa cómo se obtiene la información dinámica a visualizar.
- **Información estática a visualizar (P2_WU_SSF2):** La pregunta de la guía a partir de la que se deriva esta propiedad es también *¿Qué información se debe mostrar al usuario?* La unión de esta propiedad y la anterior da lugar al mensaje con la información a mostrar al usuario. Además de la información dinámica obtenida de la base de datos del sistema puede ser necesario mostrar algún mensaje de texto de forma estática. Esta propiedad especifica dicho mensaje.
- **Visualización del mensaje (P3_WU_SSF2):** Las preguntas de las que se ha derivado esta propiedad son *¿Qué información se debe mostrar de forma bloqueante porque es crítica?* *¿Qué información se debe resaltar porque es importante pero no crítica?* *¿Qué información se debe mostrar en el área de estado?* Estas preguntas tienen como objetivo el capturar la manera en que la información se va a visualizar al usuario, que es justamente el objetivo de esta propiedad. Mediante esta propiedad se puede especificar el formato del mensaje de texto que se muestra al usuario.

Como ejemplo para aplicar las propiedades de WU_SSF2 al sistema de alquiler de coches se ha elegido la ventana desde la que se alquila un coche, mostrada en la Figura 5.4. Desde esta ventana, el usuario podría visualizar la disponibilidad de los coches para las fechas en las cuales desea realizar el alquiler. Es decir, podría ver el estado de los coches. Así le sería más sencillo hacer la selección de modelo, color y motor. El valor de la propiedad *Información dinámica a visualizar (P1_WU_SSF2)* sería en este

caso los coches que hubiera disponibles entre las fechas introducidas por el usuario. La propiedad *Información estática a visualizar (P2_WU_SSF2)* no tiene valor en este ejemplo, ya que toda la información mostrada dependería de valores almacenados en el sistema. La propiedad *Visualización del mensaje (P3_WU_SSF2)* tomaría como valor la representación de la información en tipo tabla. Esta tabla se podría mostrar u ocultar pulsando el botón *ver disponibles* y *ocultar disponibles* respectivamente.

Alquilar un vehículo

Datos del cliente
 Cliente: Luis Pérez Número de tarjeta: 1100-34-3442788

Fechas
 Fecha de recogida: 12/02/2009 Fecha de devolución: 14/02/2009

Datos del vehículo
 Tipo de vehículo: Compacto Marca: SEAT
 Modelo: Color: Motor: Diésel Gasolina
 Ocultar disponibles

Modelo	Motor	Color	Num
León	Gasolina	Rojo	2
		Negro	1
		Azul	4
	Diésel	Rojo	3
		Negro	3
		Azul	1
		Gris	2
Ibiza	Gasolina	Rojo	1

Aceptar Cancelar

Figura 5.4 Reservar alquiler de coche donde se ve el estado de los coches

Por lo que respecta a las **propiedades de Mostrar el estado de las acciones (WU_SSF3)**, tienen como objetivo el determinar qué servicios se pueden inhabilitar y bajo qué circunstancias. Estas propiedades no se han extraído de las guías para la captura de requisitos, sino de los patrones de usabilidad de Tidwell [Tidwell, 2005]. La Tabla 5.4 muestra un resumen con

el mecanismo de usabilidad del cual se derivan las propiedades y cuál es el objetivo de cada una de ellas:

Patrón de usabilidad	Objetivo de la propiedad	Propiedad	Tipo
Disable irrelevant things	Indicar las acciones que se pueden inhabilitar en fase de ejecución	Selección de acciones (P1_WU_SSF3)	Configurable
Disable irrelevant things	Determinar la condición que al cumplirse provocará que se inhabilite la acción	Condición para inhabilitar (P2_WU_SSF3)	Configurable
Short description	Describir de forma breve la funcionalidad del sistema	Texto explicativo (P3_WU_SSF3)	Configurable

Tabla 5.4 Derivación de propiedades de la forma de uso *Mostrar el estado de las acciones* a partir de patrones de usabilidad de Tidwell

Tal como se puede apreciar en la Tabla 5.4, todas las propiedades son configurables. A continuación se explican en detalle:

- **Selección de acciones (P1_WU_SSF3):** Esta propiedad se deriva del patrón *Disable irrelevant things* propuesto por Tidwell [Tidwell, 2005], que propone inhabilitar aquellas acciones que son irrelevantes en algunos estados del sistema. De entre todas las acciones que visualiza el usuario en la interfaz, esta propiedad tiene el objetivo de seleccionar aquellas que se deben poder inhabilitar bajo cierta condición.

- **Condición para inhabilitar (P2_WU_SSF3):** Esta propiedad también se deriva del patrón *Disable irrelevant things* de Tidwell. Parte de la especificación de este patrón consiste en determinar qué acciones se consideran irrelevantes. Esta decisión puede ser dinámica y depender de una condición, dando lugar a esta propiedad. Mediante esta propiedad, el analista debe especificar la condición para inhabilitar cada una de las acciones con capacidad de inhabilitación.
- **Texto descriptivo (P3_WU_SSF3):** Esta propiedad se deriva del patrón *Short description* de Tidwell. La inhabilitación de acciones puede requerir que se explique al usuario el motivo de su inhabilitación y cómo volver a habilitarlas. Sobre todo si la inhabilitación depende de una lógica interna del sistema. Esta propiedad tiene como objetivo el permitir al analista mostrar un texto explicativo sobre el motivo de la inhabilitación de la acción. Por lo tanto es una propiedad configurable.

Como ejemplo de aplicación de estas propiedades al sistema de alquiler de coches, se ha elegido el servicio imprimir facturas. Este servicio muestra e imprime el histórico de facturas de un cliente una vez se haya identificado mediante nombre y DNI.

Datos del cliente

Cliente: DNI:

Factura creada a partir de:

Id	Fecha recogida	Fecha devolución	Importe	Modelo
A23	12/03/2008	15/03/2008	120 €	Seicento
A25	04/09/2008	07/09/2008	200 €	Polo
A34	30/10/2008	03/11/2008	250 €	Astra

Figura 5.5 Impresión de facturas donde el botón de imprimir se inhabilita si no existen facturas

En la Figura 5.5 únicamente estaría habilitado el botón *Imprimir* si el usuario hubiera seleccionado una. Al seleccionarla se habilita el botón de *Imprimir*. En este ejemplo, el valor de la propiedad *Selección de acciones (P1_WU_SSF3)* sería el servicio *imprimir facturas*, el valor de *Condición para inhabilitar (P2_WU_SSF3)* sería que el usuario no hubiera seleccionado al menos una factura y el valor de *Texto descriptivo (P3_WU_SSF3)* sería *Para imprimir factura selecciona una previamente*.

En cuanto a las **propiedades de Informar de falta de recursos (WU_SSF4)**, se utilizan para que el analista pueda especificar qué servicios avisan de la falta de recursos para su ejecución y cómo se produce este aviso. La Tabla 5.5 muestra de forma resumida las preguntas de las cuales se han extraído las propiedades y el objetivo que tiene asignado cada una de ellas.

Tal como se puede apreciar en la Tabla 5.5, todas las propiedades son configurables. A continuación se especifican en detalle estas propiedades:

- **Selección de servicios (P1_WU_SSF4):** Esta propiedad se deriva de la pregunta de la guía para la captura de requisitos, *¿Para qué servicios quiere el usuario ser informado si no existen los suficientes recursos para su ejecución?* De entre todos los servicios del sistema, el analista debe identificar aquellos que requieren un mayor consumo de recursos para aplicarles esta forma de uso. Con esta propiedad el analista especifica de entre todos los servicios que requieren más recursos, cuáles informarán de la falta de éstos (dado el caso) al solicitar su ejecución.
- **Visualización del mensaje (P2_WU_SSF4):** Las preguntas de la guía de captura de requisitos a partir de las cuales se ha derivado esta propiedad son *¿Qué información se debe mostrar de forma bloqueante porque es crítica?* *¿Qué información se debe resaltar porque es importante pero no crítica?* *¿Qué información se debe mostrar en el área de estado?* A partir de estas preguntas se deduce que es necesario una propiedad que indique cómo se mostrará el mensaje de aviso por la falta de recursos, porque la información se puede mostrar de distintas maneras, tal y como se ha comentado para la forma de uso WU_SSF1.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Para qué servicios quiere el usuario ser informado si no existen los suficientes recursos para su ejecución?	Indicar los servicios que mostrarán el mensaje de error por la falta de recursos para su ejecución	Selección de servicios (P1_WU_SSF4)	Configurable
¿Qué información se debe mostrar de forma bloqueante porque es crítica? ¿Qué información se debe resaltar porque es importante pero no crítica? ¿Qué información se debe mostrar en el área de estado?	Determinar el aspecto visual del mensaje de error por la falta de recursos para la ejecución de servicios	Visualización del mensaje (P2_WU_SSF4)	Configurable

Tabla 5.5 Derivación de propiedades de la forma de uso *Informar de falta de recursos* a partir de la guía de captura de requisitos

Como ejemplo práctico de uso de estas propiedades en el sistema de alquiler de coches, se va a utilizar la funcionalidad de reservar alquiler de coche. Supongamos que el alquiler se puede realizar desde varias oficinas (varios clientes) que trabajan sobre una misma base de datos (servidor). En este caso, si hubiera muchos clientes solicitando un alquiler, el servidor es posible que no tuviera los suficientes recursos para satisfacer la demanda

de todos los clientes. Si al solicitar la reserva de alquiler (Figura 5.4) se detectara que no existen los suficientes recursos, se mostraría un mensaje de error como el de la Figura 5.6. Para este ejemplo, la propiedad *Selección de servicios (P1_WU_SSF4)* tomaría el valor de *Reservar alquiler de coche*, y la propiedad *Visualización del mensaje (P2_WU_SSF4)* tomaría el valor de un mensaje textual de error en una ventana emergente (tal y como se puede apreciar en la Figura 5.6).

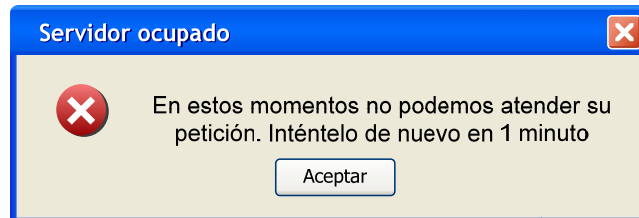


Figura 5.6 Mensaje de error por falta de recursos

Por último, las **propiedades de Informar de fallos en dispositivos externos (WU_SSF5)** indican qué servicios avisan de un fallo en la ejecución por culpa de un recurso externo y cómo se visualiza este mensaje de aviso. La Tabla 5.6 muestra de forma resumida las preguntas de las cuales se han extraído las propiedades y el objetivo que tiene asignado cada una de ellas. A continuación se explican estas propiedades en detalle:

- **Selección de servicios (P1_WU_SSF5):** La pregunta de la guía de captura de requisitos de la que se extrae esta propiedad es, *¿En qué servicios quiere el usuario ser informado cuando recursos externos que trabajan con el sistema no permitan su ejecución?* Esta pregunta tiene como objetivo el determinar bajo qué circunstancias el sistema debe avisar al usuario de los recursos externos que no funcionan correctamente. Mediante esta propiedad, el analista debe seleccionar aquellos servicios que requieren un recurso externo y que deben informar al usuario cuando el recurso externo no esté funcionando correctamente.
- **Visualización del mensaje (P2_WU_SSF5):** Las preguntas de la guía de las que se deriva esta propiedad son también las mismas, *¿Qué información se debe mostrar de forma bloqueante porque es crítica? ¿Qué información se debe resaltar porque es importante*

pero no crítica? ¿Qué información se debe mostrar en el área de estado? Por lo tanto, las alternativas de configuración de esta propiedad son las mismas que para P2_WU_SSF4.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿En qué servicios quiere el usuario ser informado cuando recursos externos que trabajan con el sistema no permitan su ejecución?	Indicar los servicios que mostrarán el mensaje de error por un mal funcionamiento de un dispositivo externo	Selección de servicios (P1_WU_SSF5)	Configurable
¿Qué información se debe mostrar de forma bloqueante porque es crítica? ¿Qué información se debe resaltar porque es importante pero no crítica? ¿Qué información se debe mostrar en el área de estado?	Determinar el aspecto visual del mensaje de error por un mal funcionamiento de un dispositivo externo	Visualización del mensaje (P2_WU_SSF5)	Configurable

Tabla 5.6 Derivación de propiedades de la forma de uso *Informar de fallos en dispositivos externos* a partir de la guía de captura de requisitos

Para ilustrar la aplicación de estas propiedades al sistema de alquiler de coches, esta tesis se ha centrado en la funcionalidad de imprimir facturas. A partir de la ventana de impresión de facturas (Figura 5.5), si el usuario pulsara el botón de imprimir y fallara la comunicación con la impresora, se mostraría el mensaje de error de la Figura 5.7. Para este ejemplo, la propiedad *Selección de servicios* ($P1_WU_SSF5$) tomaría el valor de *imprimir factura*, y la propiedad *Visualización del mensaje* ($P2_WU_SSF5$) tomaría el valor de un mensaje textual de error en una ventana emergente.

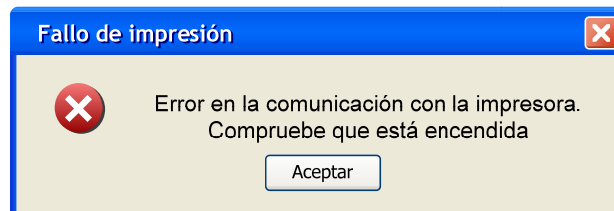


Figura 5.7 Ejemplo de mensaje de error por un fallo en la comunicación con un dispositivo externo

5.1.3 Modificación de los modelos conceptuales de OO-Method

Las cinco formas de uso implicarían cambios en OO-Method. De estas cinco, nos vamos a centrar en tres: *Informar del éxito o fracaso en la ejecución del servicio* (WU_SSF1), *Mostrar el estado de la información* (WU_SSF2) y *Mostrar el estado de las acciones* (WU_SSF3). Las otras dos formas de uso, *Informar de falta de recursos* (WU_SSF4) e *Informar de fallos en dispositivos externos* (WU_SSF5) no se van a comentar por motivos de espacio, ya que la modificación de los modelos conceptuales es muy similar a los provocados por WU_SSF1 , WU_SSF2 y WU_SSF3 .

La primera de estas formas de uso a comentar es **Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)**. Esta forma de uso ya está actualmente soportada en parte por OO-Method debido a que las aplicaciones generadas informan al usuario cuando la ejecución del servicio termina en error, pero no permite modelar los aspectos de visualización del mensaje incluidos en la propiedad *Visualización del mensaje*. Todos los mensajes son actualmente generados por medio de mensajes de texto y

este formato de visualización no se puede modificar. Además, el analista puede definir el mensaje de error que se mostrará al usuario si el error está provocado por una restricción de integridad o una precondition. En cambio, el analista no puede definir los errores provocados por intentar ejecutar un servicio desde un estado del objeto en el cual no está definido ese servicio. Para este caso, el mensaje del error es generado por el compilador de modelos de forma genérica.

A parte de estas carencias existe otra más importante, y es que el sistema no informa al usuario cuando la ejecución del servicio se ha realizado con éxito. Si el usuario quiere comprobar que ha tenido éxito la acción, debe comprobarlo él mismo a través de consultas. Estas consultas no tienen importancia si la información es visible desde la misma interfaz en la que se ha lanzado el servicio, pero si para comprobar que el servicio se ha ejecutado correctamente hay que hacer navegaciones entre interfaces, el esfuerzo del usuario para comprobar el éxito del servicio es alto. Por lo tanto, los cambios que WU_SSF1 incorpora a OO-Method para superar estos problemas son:

- El analista debe poder decidir cómo se visualizan los mensajes de éxito y de error.
- El analista debe poder introducir el mensaje de error que se mostrará al usuario cuando se produzca la solicitud de la ejecución de un servicio en un estado no válido del objeto para ese servicio.
- El compilador de modelos debe generar código para informar al usuario de los servicios que se ejecutan correctamente.
- El analista debe poder decidir el mensaje de éxito que se mostrará cuando un servicio se ejecute correctamente.

En cuanto a la forma de uso **Mostrar el estado de la información (WU_SSF2)**, también está actualmente soportada en parte por OO-Method. Los sistemas generados por OO-Method soportan la aplicación de dicha forma de uso en dos ámbitos:

1. Indicando el número de instancias mostradas en la Unidad de Interacción de Población (UIP): Al abrir una lista de instancias de objetos en las aplicaciones generadas con OO-Method, se muestra en la parte inferior derecha de la ventana el número de instancias visibles. Además, al aplicar un filtro al conjunto de instancias también se informa

al usuario con el número de instancias que cumplen la condición del filtro. En ambos casos, tanto la *información dinámica* (número de instancias), como la *información estática* (una barra “/” que indica el número de instancias visibles con respecto al total) se incorporan al sistema por medio del compilador de modelos. Por lo tanto se puede afirmar que estas dos propiedades las soporta OO-Method de forma no configurable. En cambio, no se soporta la propiedad de *Visualización del mensaje*, es decir, la forma en la que se visualiza el número de instancias no la puede decidir el analista.

2. Completando argumentos de forma automática: Los valores de ciertos argumentos en una Unidad de Interacción de Servicio (UIS) pueden depender del valor introducido previamente en otros argumentos de la misma UIS. Esta funcionalidad se representa de forma abstracta mediante el Patrón Elemental de Dependencia y el de Recuperación de Estado. Por lo tanto, parte de la aplicación de WU_SSF2 está soportada por medio de dichos Patrones Elementales. En esta aplicación sólo tiene sentido la propiedad *Información dinámica*, ya que en estos casos los argumentos dependen de valores previos y no son valores fijos (estáticos). Además, el formato del valor de los argumentos es indistinto para el propósito de ambos patrones elementales.

A parte de los dos ámbitos de aplicación de WU_SSF2 que ya están soportados actualmente por OO-Method, sería interesante añadir otro para asignar alias dinámicos a las navegaciones entre contextos. De esta forma, el alias de la navegación puede, por ejemplo, incluir el número de instancias que se encontrará el usuario al realizar la navegación, tal y como muestra la Figura 5.8 a. Así si se indica al usuario que no hay instancias en el contexto destino, se evita la navegación. También se puede añadir al alias de la navegación información sobre el contenido de la información hacia la que se navega, como por ejemplo la navegación mostrada en la Figura 5.8 b. En este ejemplo se muestra al usuario el total de facturas *pendientes* y *pagadas* que se encontrará cuando navegue a esas ventanas. Al igual que en el ejemplo anterior, en este caso se evita que el usuario tenga que realizar la navegación para obtener la información mostrada en el alias.

→ Pedidos (5)

→ Facturas (1 pend., 1 pag.)

Figura 5.8 Ejemplos de navegaciones con alias dinámicos

Resumiendo, los cambios que incorpora WU_SSF2 a OO-Method son:

- El analista debe poder definir alias dinámicos para los botones de navegación.
- El analista debe poder decidir cómo se visualizará este alias en la interfaz final.

Por último, la forma de uso **Mostrar el estado de las acciones (WU_SSF3)** no está soportada en las aplicaciones generadas con OO-Method. Esta forma de uso tiene sentido en dos tipos de acciones que puede realizar el usuario en las ventanas generadas: la ejecución de un servicio y el navegar a otro contexto. En cuanto al primer tipo, actualmente si el usuario solicita la ejecución de un servicio en una UIP que incumple una precondición o que se encuentra el objeto en un estado no válido para la ejecución del servicio, se le muestra un mensaje de error al usuario, pero no se le previene de dicho error. Siguiendo el patrón de usabilidad *Disable irrelevant things* de Tidwell [Tidwell, 2005] se mejoraría la usabilidad de las aplicaciones generadas si los botones de las acciones en una UIP se inhabilitaran en caso de que no se pudiera ejecutar el servicio. Por lo tanto se deberían poder inhabilitar los botones de acción de las Unidades de Interacción de Instancia (UIIs) y Unidades de Interacción de Población (UIPs) que no se puedan ejecutar en el estado actual del objeto o por alguna precondición. Además, también se debería poder inhabilitar el botón de la UIS que lanza la ejecución del servicio cuando se incumpla alguna precondición dependiente de argumentos de entrada que el usuario haya rellenado en esa UI.

Por otro lado, actualmente los botones de navegación no se inhabilitan aunque la navegación hacia el contexto destino no muestre ninguna información. La única forma de descubrir si la navegación devuelve información es ejecutándola. En algunos casos, la navegación hacia un contexto vacío tiene sentido si en el contexto destino se pueden crear instancias. En cambio, si el contexto destino solo permite la visualización o modificación de instancias (no la creación), la navegación sólo supone una pérdida de tiempo para el usuario.

Los cambios que WU_SSF3 incorpora a OO-Method afecta a inhabilitar servicios y navegaciones:

- Inhabilitar servicios: El compilador de modelos, sin la ayuda del analista, debe incluir en el sistema la capacidad de inhabilitar aquellos servicios que no se pueden ejecutar por incumplirse una precondición o estar el objeto en un estado desde el cual el servicio no se puede ejecutar. Por lo tanto, la propiedad configurable *Definición de la condición* está ya representada en las precondiciones y en el Modelo Dinámico. En cuanto a la propiedad *Selección de acciones*, es no configurable en OO-Method a pesar de estar definida como configurable. Se trata de una propiedad establecida, ya que en OO-Method tiene sentido que todos los servicios con precondiciones o que sólo se puedan ejecutar en ciertos estados del objeto se inhabiliten cuando no se puedan lanzar, independientemente de la decisión del analista. Por último, la propiedad *Texto descriptivo* está ya soportada por los mensajes de error que se lanzan cuando un servicio se intenta ejecutar incumpliendo una precondición o desde un estado del objeto no válido (propiedad *Visualización del mensaje* de la forma de uso *Informar del éxito o fracaso en la ejecución del servicio*).
- Inhabilitar navegaciones: El analista debe poder decidir qué navegaciones desea inhabilitar cuando se navegue a contextos vacíos. Por lo tanto, al aplicar WU_SSF3 a OO-Method, la única propiedad configurable es la de *Selección de acciones*, ya que la propiedad *Condición para inhabilitar* tendrá siempre el mismo valor, que no haya instancias en el contexto destino. Por tanto, esta última propiedad es un caso más de propiedad establecida ya que se definió como configurable, pero a la hora de aplicarla a OO-Method las posibles condiciones para inhabilitar las navegaciones son muy limitadas y por tanto, la definición de dichas condiciones se puede delegar al compilador de modelos, ganando eficiencia en el desarrollo de sistemas. La propiedad *Texto descriptivo* no tiene sentido aplicarla a la inhabilitación de navegaciones porque todas las inhabilitaciones serían por el mismo motivo: el contexto destino no tiene instancias.

La Tabla 5.7 muestra de forma resumida los cambios que se incorporarían a OO-Method a partir de cada una de las propiedades.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)	Selección de servicios (P1_WU_SSF1)	No configurable	Los servicios deben informar cuando se ejecutan con éxito
	Visualización del mensaje (P2_WU_SSF1)	Configurable	<ul style="list-style-type: none"> – El analista debe decidir cómo visualizar los mensajes (error y éxito) – El analista debe decidir el texto de los mensajes de error al ejecutar un servicio en un estado no válido – El analista debe decidir el contenido de los mensajes de éxito
Mostrar el estado de la información (WU_SSF2)	Información dinámica a visualizar (P1_WU_SSF2)	Configurable	Se deben poder definir alias dinámicos para las navegaciones
	Información estática a visualizar (P2_WU_SSF2)	Configurable	No implica cambios

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
	Visualización del mensaje (P3_WU_SSF2)	Configurable	El analista debe poder decidir cómo se visualizará el alias (dinámico y estático) en las navegaciones
Mostrar el estado de las acciones (WU_SSF3)	Selección de acciones (P1_WU_SSF3)	Establecida (configurable para las navegaciones, no configurable para acciones)	<ul style="list-style-type: none"> – Posibilidad de inhabilitar los servicios y navegaciones – El analista debe poder seleccionar las navegaciones a inhabilitar
	Condición para inhabilitar (P2_WU_SSF3)	Establecida (configurable para las acciones, no configurable para las navegaciones)	El compilador de modelos debe añadir la condición para inhabilitar las navegaciones (que no existan instancias en el contexto destino)
	Texto descriptivo (P3_WU_SSF3)	Configurable	No implica cambios



Tabla 5.7 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *System Status Feedback*

En la siguiente sección se detallan los cambios a nivel de modelado conceptual que es necesario llevar a cabo para la incorporación de estas formas de uso a OO-Method.


5.1.3.1 WU_SSF1: Informar del éxito o fracaso en la ejecución del servicio

La única propiedad configurable de WU_SSF1, *Visualización del mensaje (P2_WU_SSF1)*, es la que enriquece el modelado conceptual con más primitivas. Mediante esta propiedad, el analista puede establecer la manera en que se mostrará la información al usuario. Esta propiedad configura tanto la visualización de los mensajes de error, como la de los mensajes de éxito en la ejecución. Las nuevas primitivas se han definido en dos modelos: Modelo de Interacción Concreto y Modelo de Objetos.

Por un lado, en el **Modelo de Interacción Concreto**, el analista debería usar las nuevas primitivas para dos casos: los mensajes que se mostrarán si la ejecución ha sido correcta y los que se mostrarán en caso de error. El modelado de los mensajes de error en este modelo incluye los provocados por: incumplir una precondición; incumplir una restricción de integridad; ejecutar un servicio en un estado del objeto desde el cual no está permitida su ejecución. Existen distintas primitivas para configurar las posibles variantes de visualización:

- Mostrar información mediante texto:
 - En una nueva ventana:
 - Bloqueante (modal) o no bloqueante
 - Tipo de Mensaje: aviso, error o alerta
 - Fuente del texto
 - Tamaño
 - Color
 - Alineación
 - Texto en alguna parte de la ventana principal:
 - Ubicación en la ventana principal
 - Fuente del texto
 - Tamaño
 - Color
 - Alineación
- Mostrar información mediante iconos:
 - Icono a mostrar, por ejemplo  para cuando la acción se ejecute correctamente y  cuando haya algún fallo
 - Dónde mostrar el icono dentro de la ventana principal

- No mostrar ninguna información.

Por defecto, los mensajes que avisen del éxito en la ejecución de un servicio se mostrarán mediante el icono  en la esquina inferior derecha de la ventana principal. En cuanto a los mensajes de error, se mostrarán en una ventana modal de tipo error con fuente Arial, tamaño 10, color negro y alineación centrada.

Por otro lado, el **Modelo de Objetos** se debe enriquecer con dos Primitivas mediante las cuales se pueden definir los mensajes textuales. Estas primitivas solo se tienen en cuenta en el caso de que la información de estado del servicio se muestre mediante texto:

- Mensaje de error por una transición no válida entre estados: para cada uno de los servicios del Modelo de Objetos, el analista puede introducir un mensaje de error que se mostrará al usuario cuando éste intente ejecutar un servicio desde un estado del objeto en el que esta ejecución no esté permitida. Si no se introduce ningún mensaje, la aplicación puede mostrar uno genérico. En el resto de mensajes de error (por incumplir restricción de integridad o precondition) el analista ya puede actualmente definir el texto a mostrar, por lo que no hay que incorporar primitivas nuevas para esta funcionalidad.
- Mensaje de éxito: mediante esta primitiva, el analista decide el texto que se mostrará cuando el servicio se ejecute correctamente para cada uno de los servicios. Si no se introduce ningún texto, la aplicación mostrará uno genérico creado por el propio compilador de modelos de OO-Method.

La Tabla 5.8 muestra de forma resumida los cambios que incorpora esta forma de uso al modelado conceptual de OO-Method.

Modelo de OO-Method	Nueva primitiva conceptual
Modelo de Interacción Concreto	– Tipo de visualización: en texto, gráficamente u oculta – Formato del texto y de la ventana donde se muestra la información – Selección de iconos y ubicación de éstos en el sistema

Modelo de Objetos	<ul style="list-style-type: none"> – Texto del mensaje de error por causa de la ejecución de un servicio en un estado no válido – Texto del mensaje de éxito
-------------------	--

Tabla 5.8 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Informar del éxito o fracaso en la ejecución del servicio*

5.1.3.2 WU_SSF2: Mostrar el estado de la información

Tal y como se ha comentado anteriormente, la aplicación de esta forma de uso sobre OO-Method deriva en la incorporación de alias dinámicos para las navegaciones incluidas en las UIPs y UIIs. Las tres propiedades que componen esta forma de uso son configurables y por tanto necesitan de Primitivas Conceptuales para representarlas de forma abstracta. Sin embargo, la propiedad *Información estática a visualizar (P2_WU_SSF2)* ya tiene actualmente una primitiva conceptual en el Modelo de Interacción Abstracto y por tanto no requiere añadir nuevas primitivas para su representación. La primitiva para representar la *Información estática a visualizar* equivale al alias del Patrón Elemental de Navegación que se representa en el Modelo de Interacción Abstracto actualmente.

Por lo tanto, sólo implican cambios las propiedades *Información dinámica a visualizar (P1_WU_SSF2)* y *Visualización del mensaje (P3_WU_SSF2)*. Por un lado, *Información dinámica a visualizar* se representará en el **Modelo de Interacción Abstracto** mediante la primitiva conceptual *Fórmula para definir el alias dinámico en las navegaciones*. Esta primitiva se usará para definir la fórmula mediante la cual se especifica la información dinámica que se va a mostrar en el alias. La fórmula se definirá en base a la combinación de los siguientes elementos:

- La instancia seleccionada en el actual contexto (variable THIS)
- Valores de atributos de las clases que componen el Modelo de Objetos
- Funciones estándar para trabajar con fechas, números y cadenas.
- Funciones de usuario
- Operadores aritméticos, operadores de colección, constantes, operadores lógicos y operadores con cadenas.

Si el analista no define ningún alias dinámico, por defecto no existirá ninguno.

Por otro lado, la propiedad *Visualización del mensaje (P3_WU_SSF2)* se representará en el **Modelo de Interacción Concreto** ya que está relacionado con el cómo se muestra la información en la interfaz. Se debe añadir una primitiva por cada uno de los aspectos visuales que se puedan configurar:

- Formato del texto del botón de navegación
- Tamaño del texto del botón de navegación
- Tamaño del botón de navegación
- Color del botón de navegación
- Icono del botón de navegación

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Abstracto	Fórmula con la que se obtiene el alias dinámico
Modelo de Interacción Concreto	Opciones de visualización sobre el formato del alias y el aspecto del botón de navegación

Tabla 5.9 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Mostrar el estado de la información*

Por defecto, el formato será letra Arial de tamaño 12 y el tamaño del botón será el necesario para introducir su alias.

La Tabla 5.9 muestra en resumen el cambio que incorpora esta forma de uso en el modelo conceptual.

5.1.3.3 WU_SSF3: Mostrar el estado de las acciones

La incorporación de esta forma de uso a OO-Method implica inhabilitar los servicios cuya ejecución no está permitida en el estado actual del sistema e inhabilitar las navegaciones hacia contextos donde no hay ninguna información que mostrar. Las propiedades de esta forma de uso son las dos configurables, pero el modelado se lleva a cabo de forma distinta

dependiendo de si el elemento a inhabilitar es una acción o una navegación. Por lo tanto, vamos a separar las dos instanciaciones sobre OO-Method que tiene esta forma de uso, ya que cada una de ellas da lugar a primitivas conceptuales distintas: inhabilitación de acciones e inhabilitación de navegaciones. Por un lado, para el caso de la **inhabilitación de acciones**, WU_SSF3 tiene tres propiedades configurables, *Selección de acciones* (*P1_WU_SSF3*), *Condición para inhabilitar* (*P2_WU_SSF3*) y *Texto descriptivo* (*P3_WU_SSF3*). En cambio, ninguna de estas propiedades implica cambios a nivel conceptual en OO-Method. Todas las acciones que se pueden definir en OO-Method son sensibles a poderse inhabilitar, por lo tanto esta propiedad se instancia en no configurable al aplicarla a OO-Method. Es decir, es una propiedad establecida. El compilador de modelos lo aplicará a todas las acciones de forma automática sin la intervención del analista.

En cuanto a la otra propiedad configurable, *Condición para inhabilitar*, ya existen primitivas conceptuales para representar las condiciones que se deben dar para inhabilitar las acciones. Estas condiciones se extraen de: (1) las precondiciones asociadas a la ejecución de los servicios; (2) la comprobación de si el estado del objeto desde el que se lanza el servicio es un estado válido para la ejecución de ese servicio. Según estas condiciones se inhabilitarán las acciones cuya precondición fuera falsa o cuya ejecución no esté definida para el estado actual del objeto respectivamente. Por lo tanto esta propiedad tampoco incorpora modificaciones en el modelo conceptual de OO-Method.

Por último, la propiedad *Texto descriptivo* tampoco implica cambios a nivel conceptual porque ya existen Primitivas Conceptuales que determinan los mensajes de las precondiciones y cuando se intenta ejecutar un servicio desde un estado del objeto no válido.

Por otro lado, para el caso de **inhabilitar navegaciones**, de las tres propiedades configurables de WU_SSF3, la única que implica cambios a nivel conceptual al instanciarla a OO-Method es la de *Selección de acciones* (*P1_WU_SSF3*). Dependiendo de los requisitos del usuario, puede ser interesante permitir navegaciones a contextos vacíos o no, ya que en algunos contextos se pueden crear instancias a pesar de no contener ninguna información. Por lo tanto, es necesaria una primitiva conceptual que

indique qué navegaciones se pueden inhabilitar y cuales no. Esta primitiva se debe añadir al **Modelo de Interacción Abstracto**, ya que es una primitiva relacionada con el qué se visualiza del sistema.

La propiedad que indica cuándo se deben inhabilitar las navegaciones, *Condición para inhabilitar (P2_WU_SSF3)*, es no configurable a la hora de instanciarla a OO-Method porque la condición para inhabilitar es que el número de instancias del contexto destino sea igual a 0 para todos los sistemas desarrollados. Es decir, se trata de una propiedad establecida. Por lo tanto, esta propiedad no necesita representación conceptual y la responsabilidad de darle valor recae únicamente sobre el compilador de modelos.

Por último, la propiedad *Texto descriptivo (P3_WU_SSF3)* no tiene sentido el aplicarla a las navegaciones de OO-Method porque la explicación para inhabilitar las navegaciones es siempre la misma: que el contexto destino no contenga instancias.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Abstracto	Indicar las Navegaciones que se pueden inhabilitar

Tabla 5.10 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Mostrar el estado de las acciones*

Por defecto no habrá ninguna navegación con la capacidad de inhabilitarse. Es el analista el que debe seleccionar aquellas navegaciones con capacidad de inhabilitación.

La Tabla 5.10 muestra en resumen el único cambio que incorpora esta forma de uso al modelado conceptual de OO-Method:

Una vez detectados los cambios que se deberán realizar en los modelos conceptuales de OO-Method, el próximo paso es el de detectar los cambios a incorporar en el compilador de modelos. La siguiente sección muestra estos cambios en detalle.

5.1.4 Modificación del compilador de modelos

5.1.4.1 WU_SSF1: Informar del éxito o fracaso en la ejecución del servicio

Tanto las propiedades configurables como las no configurables de la forma de uso *Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)* implican cambios en el compilador de modelos. La Figura 5.9 muestra el Diagrama de Clases con las clases y métodos que implementan las formas de uso *WU_SSF1* y *Mostrar el estado de las acciones (WU_SSF3)*.

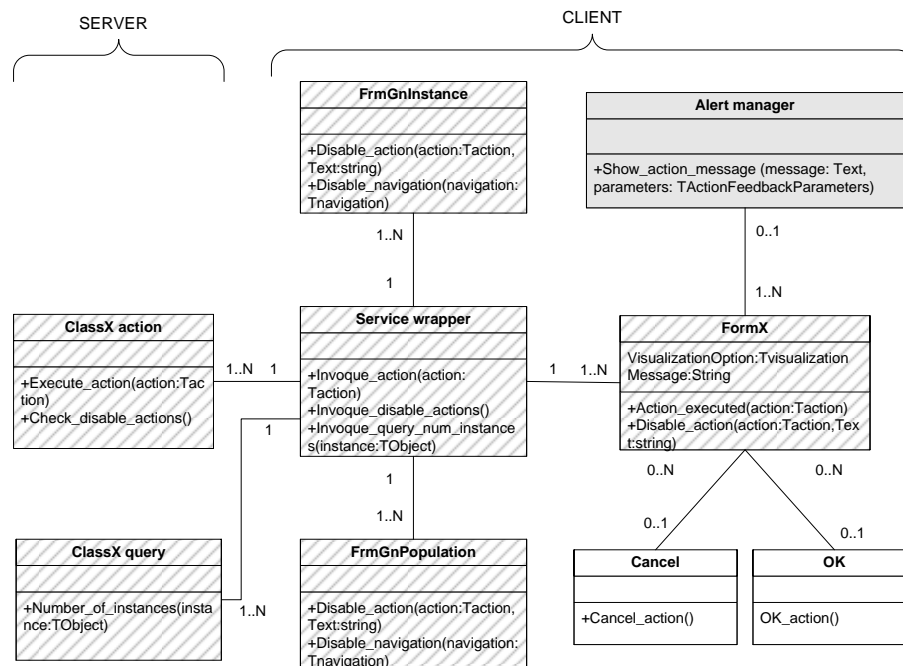


Figura 5.9 Diagrama de Clases para *WU_SSF1* y *WU_SSF3* de *System Status Feedback*

Para dar soporte a *WU_SSF1*, la única clase que se añade nueva es *Alert manager*, que implementa cómo se lleva a cabo la visualización del mensaje sobre el estado del sistema. Las clases *OK* y *Cancel* no se ven afectadas por este mecanismo de usabilidad. Por último, hay que modificar la clase

Form X. Esta clase debe añadir un método para invocar a la clase *Alert manager* que informará al usuario sobre el éxito o el fracaso en la ejecución por medio de un mensaje. Además, se debe añadir un atributo para almacenar las opciones de visualización de ese mensaje. El resto de clases de la Figura 5.9 son clases que se ven afectadas por *WU_SSF3* y se comentarán en la sección correspondiente.

La Figura 5.10 muestra en detalle cómo se producirá la invocación de métodos entre las clases que implementan el patrón. Si el usuario ejecutara un servicio mediante la invocación a *OK_action*, la clase *Service wrapper* lanza la petición a la clase encargada de ejecutar el servicio (el método encargado de esta labor ya existía previamente, por eso la clase está coloreada en blanco). Esta clase se llama *ClassX action* y se encuentra en la parte servidora. Una vez que el servicio se haya ejecutado, el usuario puede ver su resultado gracias a la clase *Alert manager* que mostrará un mensaje indicando el éxito o el fracaso en la ejecución del servicio. La visualización de este mensaje se hará con las opciones de visualización que haya modelado el analista.

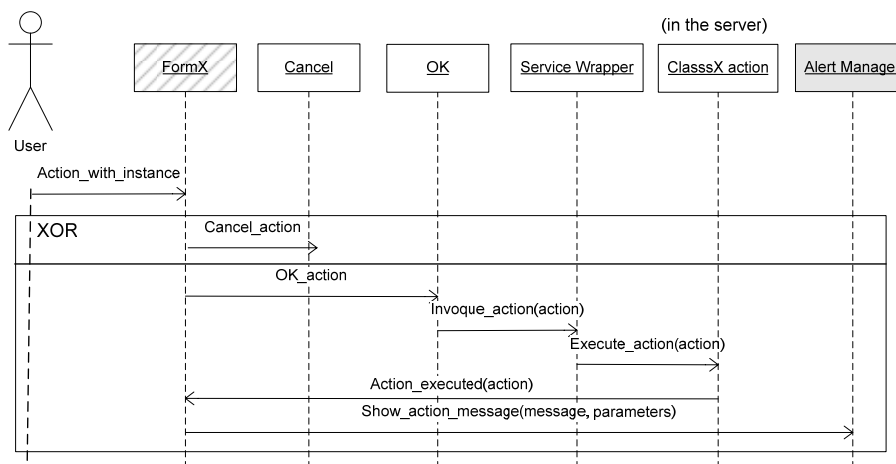


Figura 5.10 Diagrama de Secuencia para incorporar la forma de uso *Informar del éxito o fracaso en la ejecución del servicio*

La Tabla 5.11 muestra un resumen de los cambios que incorpora *Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)* al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	Invoca a la clase <i>Alert manager</i> para mostrar el estado del sistema
Alert Manager	<ul style="list-style-type: none"> – Esta nueva clase almacena las opciones de visualización del mensaje que contiene la información a visualizar – Es la encargada de visualizar la información

Tabla 5.11 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Informar del éxito o fracaso en la ejecución del servicio*

5.1.4.2 WU_SSF2: Mostrar el estado de la información

Para representar los cambios que hay que llevar a cabo en el compilador de modelos hemos usado un Diagrama de Clases distinto al utilizado para explicar los cambios provocados por WU_SSF1. Aunque todos los cambios que provoca cada uno de los mecanismos de usabilidad se pueden representar en un mismo Diagrama de Clases, en este caso hemos decidido separar el Diagrama de Clases entre las dos formas de uso porque las clases afectadas por la implementación de ambas son totalmente distintas.

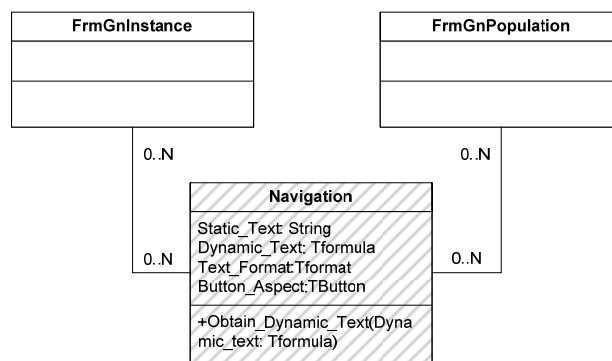


Figura 5.11 Diagrama de Clases para la WU_SSF2 de *System Status Feedback*

El Diagrama de Clases de la Figura 5.11 muestra la única clase afectada por la forma de uso WU_SSF2: *Navigation*. Esta clase tiene un atributo para

representar el texto estático de la navegación (*Static_Text*), otro para representar la fórmula que proporciona el valor al texto dinámico (*Dynamic_Text*), otro para representar el tipo de formato para el texto (*Text_Format*) y otro para representar las opciones de visualización del botón de navegación (*Button_Aspect*). El formato del texto es el mismo para la parte estática y la dinámica, ya que la etiqueta de la navegación es la combinación de ambos.

La secuencia de llamadas entre métodos para generar la etiqueta del botón de navegación está representada en la Figura 5.12. Al abrir el usuario la ventana (generada a partir de una UIP o una UII), mediante el método *Obtain_Dynamic_and_Static_Text* se obtendrá la parte la etiqueta del botón de navegación juntando la etiqueta dinámica y la estática. Esta etiqueta se dibujará en la ventana con el método *Show_navigation_alias*.

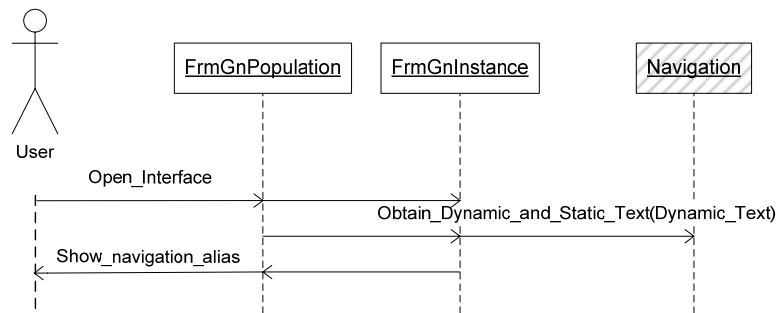


Figura 5.12 Diagrama de Secuencia para incorporar la forma de uso *Mostrar el estado de la información*

La Tabla 5.12 muestra un resumen de los cambios que incorpora WU_SSF2 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Navigation	Esta clase almacena la parte del alias estático, la fórmula para obtener el alias dinámico y cómo se visualizará la unión de ambos

Tabla 5.12 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Mostrar el estado de la información*

5.1.4.3 WU_SSF3: Mostrar el estado de las acciones

Los cambios que incorpora esta forma de uso están representados en el Diagrama de Clases de la Figura 5.9 junto con los de WU_SSF1. Los cambios por clase que incorpora WU_SSF3 son los siguientes:

- *FrmGnInstance* y *FrmGnPopulation* deben añadir métodos para solicitar la inhabilitación de servicios y navegaciones. Estos métodos se han llamado *Disable_action* y *Disable_navigations* respectivamente.
- *Service wrapper* debe añadir un método para que la clase *Class X action* verifique las acciones a inhabilitar, llamado *Invoque_disable_actions*. Además, esta clase debe añadir otro método llamado *Invoque_query_num_instances* para que la clase *Class X query* cuente el número de instancias en el contexto objetivo de la navegación.
- *Class X action* debe añadir un método llamado *Check_disable_actions* para verificar las acciones que se deben inhabilitar.
- *Class X query* debe añadir un método llamado *Number_of_instances* para contar el número de instancias en el contexto objetivo de una navegación.

En la Figura 5.13 se representa la secuencia en la invocación de los nuevos métodos. Cuando el usuario abra la interfaz de una UIP o una UII, se comprobará si hay algunos servicios que se deban inhabilitar, bien porque no se cumpla su precondición o porque su ejecución no esté permitida en el estado actual del objeto. Además, al abrir una UIS se debe comprobar si el valor introducido por el usuario en los argumentos de entrada no incumple alguna precondición. Estas comprobaciones se llevarán a cabo mediante los métodos *Invoque_disable_actions* y *Check_disable_actions*. Aquellos servicios que se deban inhabilitar se inhabilitarán mediante el método *Disable_action* que además mostrará un texto describiendo porqué se ha inhabilitado el servicio. Por último, también es necesario inhabilitar aquellas navegaciones de las UIPs y UIIs en las que el analista haya indicado que se inhabilitaran en caso de que naveguen a contextos vacíos. La comprobación

de si en el contexto destino existe alguna instancia relacionada con la seleccionada en el contexto origen se llevará a cabo mediante los métodos *Invoque_query_num_instances* y *Number_of_instances*. Ambos métodos llevan como parámetro la instancia sobre la que se aplica la navegación.

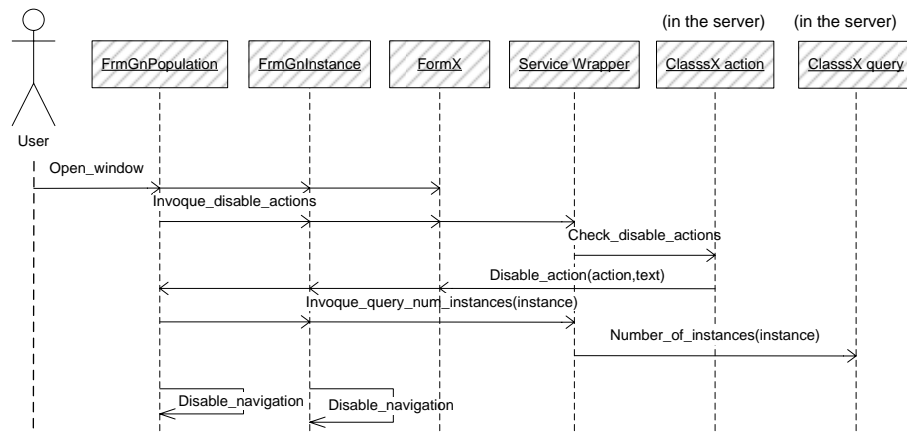


Figura 5.13 Diagrama de Secuencia para incorporar la forma de uso *Mostrar el estado de las acciones*

La Tabla 5.13 muestra un resumen de los cambios que incorpora *Mostrar el estado de las acciones (WU_SSF3)* al compilador de modelos.

Clase Genérica	Cambios que incorpora
FrmGnPopulation	<ul style="list-style-type: none"> – Tiene la posibilidad de inhabilitar acciones que hay dentro de la UIP – Puede consultar el número de instancias de los contextos destino de las navegaciones – Puede inhabilitar navegaciones si el contexto destino no tiene instancias
FrmGnInstance	<ul style="list-style-type: none"> – Tiene la posibilidad de inhabilitar acciones que hay dentro de la UII – Puede consultar el número de instancias de los contextos destino de las navegaciones – Puede inhabilitar navegaciones si el contexto destino no tiene instancias

Clase Genérica	Cambios que incorpora
Service Wrapper	<ul style="list-style-type: none"> – Consulta al servidor las precondiciones y el estado del DTE para saber qué acciones inhabilitar – Consulta al servidor el número de instancias del contexto destino de la navegación
Class X Action	Comprueba las precondiciones y el estado del Diagrama de Transición entre Estados para saber qué acciones inhabilitar
Class X Query	Comprueba el número de instancias del contexto destino de la navegación

Tabla 5.13 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Mostrar el estado de las acciones*

5.1.4.4 Resumen de los cambios que provoca System Status Feedback

La Tabla 5.14 muestra de manera resumida las formas de uso extraídas del mecanismo de usabilidad *System Status Feedback*, las propiedades que componen las formas de uso, los modelos que se deben modificar, las nuevas primitivas conceptuales que se deben introducir en dicho modelos y los cambios que hay que hacer en el compilador de modelos para implementar su funcionalidad.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Informar del éxito o fracaso en la ejecución del servicio	Selección de servicios	Ninguno	Ya soportada	Ninguno
	Visualización del mensaje	Objetos	<ul style="list-style-type: none"> – Texto del mensaje de error – Texto del mensaje de éxito 	La clase <i>Form X</i> debe almacenar el texto

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
		Interacción Concreto	<ul style="list-style-type: none"> - Tipo de visualización (texto, gráfico u oculta) - Formato del texto y de la ventana - Selección de iconos y su ubicación 	<ul style="list-style-type: none"> - La clase <i>FormX</i> debe almacenar las opciones de visualización - La clase <i>Alert manager</i> visualizará el mensaje
Mostrar el estado de la información	Información dinámica a visualizar	Modelo de Interacción Abstracto	Fórmula que define el alias dinámico	<i>Navigation</i> almacenará la fórmula del alias dinámico
	Información estática a visualizar	Ninguno	Ya soportada	Ninguno
	Visualización del mensaje	Modelo de Interacción Concreto	Aspectos visuales del texto del alias y del botón de navegación	<i>Navigation</i> almacenará las opciones de visualización del texto y del botón de navegación
Mostrar el estado de las acciones	Selección de acciones	Modelo de Interacción Abstracto	Indicar navegaciones que se puedan inhabilitar	<ul style="list-style-type: none"> - <i>FrmGnPopulation</i> y <i>FrmGnInstance</i> pueden deshabilitar acciones y navegaciones, consultar instancias destino de navegaciones

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
				<ul style="list-style-type: none"> – <i>Service wrapper</i> invoca la consulta de estado del DTE y precondiciones para saber lo que inhabilitar – <i>Class X action</i> comprueba precondición y DTE – <i>Class X query</i> calcula el número de instancias a las que navega
	Condición para inhabilitar	Ninguno	Ya soportada	Ninguno
	Texto descriptivo	Ninguno	Ninguno	El texto se muestra junto al botón inhabilitado
Informar de falta de recursos	Selección de servicios	Ninguno	Ninguno	Ninguno
	Visualización del mensaje	Ninguno	Ninguno	Ninguno
Informar de fallos en dispositivos externos	Selección de servicios	Ninguno	Ninguno	Ninguno
	Visualización del mensaje	Ninguno	Ninguno	Ninguno

Tabla 5.14 Resumen de los cambios para incorporar *System Status Feedback*

5.2 Interaction Feedback

Este mecanismo de usabilidad se introduce en los sistemas para que el usuario sea informado de que sus peticiones de interacción han sido recibidas. Cada vez que el usuario realiza una interacción con el sistema, como un movimiento del ratón o pulsar una tecla, se le debe informar de que esta interacción ha sido recibida. Los sistemas de hoy en día informan cada vez que el usuario mueve el ratón o pulsa una tecla gracias al sistema operativo sobre el que se ejecuta el sistema. Es decir, este tipo de información no depende del sistema, sino del sistema operativo. Por lo tanto, nos vamos a centrar sólo en confirmar las peticiones del usuario sobre la ejecución de un servicio, cuya confirmación sí que depende del sistema.

5.2.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *Interaction Feedback* (Anexo I) derivamos una única forma de uso, **Informar que la interacción está siendo atendida (WU_IF1)**. La Tabla 5.15 muestra de forma resumida la pregunta de la guía de captura de requisitos de la que se deriva la forma de uso y el objetivo que se pretende conseguir con ella.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Para qué acciones quiere el usuario recibir retroalimentación indicando que su petición está siendo atendida?	Informar al usuario de que el sistema está procesando la petición de ejecución de un servicio	Informar que la interacción está siendo atendida (WU_IF1)

Tabla 5.15 Derivación de la forma de uso de *Interaction Feedback* a partir de la guía de captura de requisitos

La pregunta de la guía para la captura de requisitos de *Interaction Feedback* a partir de la cual se ha derivado esta forma de uso es, *¿Para qué acciones*

quiere el usuario recibir retroalimentación indicando que su petición está siendo atendida? El objetivo de esta pregunta es el mismo que el de WU_IF1, es decir, dotar al sistema de la funcionalidad necesaria para informar al usuario que su petición de ejecución de un servicio se está ejecutando. El sistema debería proporcionar retroalimentación al usuario para confirmarle que el evento ha sido registrado. Los autores que proponen esta aplicación del mecanismo de usabilidad *Interaction Feedback* son Brighton, mediante el patrón de usabilidad *Interaction Feedback* [Griffiths, 2002] y Coram, mediante el patrón de interacción *Modelling Feedback Area* [Coram, 1996].

Por ejemplo, en el sistema de alquiler de coches, cada vez que el usuario pulse el servicio que ejecuta la reserva del alquiler, el puntero del ratón cambia a un reloj de arena para informar al usuario de que se está procesando su petición.

5.2.2 Propiedades

Cada una de las maneras que tiene el sistema de informar al usuario que su petición de ejecución sobre un servicio está siendo atendida da lugar a una propiedad. Las **propiedades de informar que la interacción está siendo atendida (WU_IF1)** se utilizan para determinar de qué manera informará el sistema al usuario que su petición de ejecución de un servicio está siendo atendida. La Tabla 5.16 muestra una visión global de las preguntas de la guía de captura de requisitos de las que se han derivado las propiedades y los objetivos de esas propiedades.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Para cada acción que se solicite se va a reconocer su petición mediante los botones?	Indicar al usuario que su petición de ejecución está siendo atendida resaltando el botón del servicio	Marcar un botón (P1_WU_IF1)	No configurable

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Para cada acción que se solicite se va a reconocer la petición oscureciendo elementos de la ventana?	Indicar al usuario que su petición de ejecución está siendo atendida inhabilitando los campos editables de la interfaz	Inhabilitar componentes de la interfaz (P2_WU_IF1)	No configurable
¿Para cada acción que se solicite se va a reconocer su petición cambiando la forma del cursor?	Indicar al usuario que su petición de ejecución está siendo atendida cambiando el puntero del ratón	Cambiar el puntero del ratón (P3_WU_IF1)	No configurable

Tabla 5.16 Derivación de propiedades de la forma de uso *Informar que la interacción está siendo atendida* a partir de la guía de captura de requisitos

A continuación se presenta en detalle cada una de las propiedades de WU_IF1:

- **Marcar un botón (P1_WU_IF1):** La pregunta de la guía de captura de requisitos a partir de la cual se ha obtenido esta propiedad es, *¿Para cada acción que se solicite se va a reconocer la petición mediante los botones?* El objetivo de esta pregunta es determinar si se va a informar al usuario de que su petición de ejecución está siendo atendida resaltando el botón que se ha pulsado. Cada vez que el usuario pulse un botón, éste debería permanecer pulsado mientras el sistema ejecuta la acción correspondiente. Esta propiedad es no configurable, ya que debería estar presente en todos los botones del sistema [Griffiths, 2002] [Coram, 1996]. La facilidad de implementación y la poca sobrecarga que añade al

funcionamiento del sistema favorece que se incorpore a todos los sistemas.

- **Inhabilitar componentes de la interfaz (P2_WU_IF1):** La pregunta de la guía a partir de la cual se deriva esta propiedad es, *¿Para cada acción que se solicite se va a reconocer la petición oscureciendo algunos elementos de la ventana?* El objetivo de esta pregunta es determinar si se va a informar al usuario de que su petición de ejecución está siendo atendida inhabilitando los campos editables de la ventana. Cada vez que el usuario pulse un botón de una ventana se deberían inhabilitar los botones y el resto de componentes de la interfaz mientras el sistema esté ejecutando la acción asociada a ese botón. El inhabilitar los componentes implica, además de que el usuario no pueda usarlos, que éstos cambien a color gris y de esta forma el usuario puede percibir que su petición está siendo atendida. Esta propiedad es no configurable, ya que debería estar presente en todas las interfaces del sistema [Griffiths, 2002] [Coram, 1996].
- **Cambiar el puntero del ratón (P3_WU_IF1):** La pregunta de la guía a partir de la cual se deriva esta propiedad es, *¿Para cada acción que se solicite se va a reconocer la petición cambiando la forma del cursor?* El objetivo de esta pregunta es determinar si se va a informar al usuario de que su petición de ejecución está siendo atendida cambiando el puntero del ratón. Por ejemplo, en entornos Windows, la flecha del ratón cambia a reloj de arena mientras el sistema está procesando una petición. Esta propiedad es no configurable ya que debería estar presente en todos los sistemas [Griffiths, 2002] [Coram, 1996].

Como ejemplo de aplicación de estas propiedades sobre el sistema de alquiler de coches, se va a utilizar el servicio reservar alquiler de coche. Tal y como se puede apreciar en la Figura 5.14, cuando el usuario solicitase la ejecución del servicio, el botón que solicita el servicio permanecería inhabilitado junto con el resto de campos editables de la ventana y el puntero del ratón cambiaría a reloj de arena. Para ello, la propiedad *Marcar un botón (P1_WU_IF1)* tomaría el valor verdadero para el botón *Aceptar* de la interfaz, la propiedad *Inhabilitar componentes de la interfaz (P2_WU_IF1)* tomaría el valor verdadero, y la propiedad *Cambiar el puntero del ratón*

(*P3_WU_IF1*) tomaría el valor verdadero. Toda esta funcionalidad se incorporaría al sistema de forma automática sin la decisión del analista, ya que las tres propiedades que componen esta forma de uso son todas no configurables.

Figura 5.14 Reservar alquiler de coche con inhabilitación de componentes de la interfaz

5.2.3 Modificación de los modelos conceptuales de OO-Method

Las aplicaciones generadas con OO-Method ya soportan la propiedad *Cambiar el puntero del ratón* (*P3_WU_IF1*). En cambio, las propiedades *Marcar un botón* (*P1_WU_IF1*) e *Inhabilitar componentes de la interfaz* (*P2_WU_IF1*) no están aún soportadas. Por lo tanto, los cambios que hay que añadir a OO-Method pretenden conseguir que:

- El sistema generado resalte el botón que se ha pulsado para lanzar el servicio mientras dure su ejecución.
- El sistema generado inhabilite los campos editables de la interfaz mientras dure la ejecución del servicio lanzado por el usuario.

La Tabla 5.17 muestra en resumen los cambios que incorporan las propiedades de la única forma de uso del mecanismo de usabilidad *Interaction Feedback*.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Informar que la interacción está siendo atendida (WU_IF1)	Marcar un botón (P1_WU_IF1)	No configurable	El sistema generado debe resaltar el botón que pulse el usuario mientras dure su ejecución
	Inhabilitar componentes de la interfaz (P2_WU_IF1)	No configurable	El sistema generado debe inhabilitar los campos editables de la interfaz desde la que lance la ejecución del servicio

Tabla 5.17 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Interaction Feedback*

Estas propiedades son no configurables y, por lo tanto, para añadirlas a OO-Method no es necesario modificar los modelos conceptuales, sino sólo el compilador de modelos. La siguiente sección explica en detalle los cambios necesarios para incorporar este patrón a OO-Method.

5.2.4 Modificación del compilador de modelos

5.2.4.1 WU_IF1: Informar que la interacción está siendo atendida

De todos los tipos de UIs que hay en el Modelo de Interacción de OO-Method, el único que se ve afectado por el mecanismo de usabilidad

Interaction Feedback es la UI de Servicio, que representa el único tipo de interfaces desde las que se puede lanzar la ejecución de una acción. El Diagrama de Clases de la Figura 5.15 muestra cómo la única clase afectada por este patrón es *FormX*, que es la clase que implementa una UI de Servicio. Los métodos que se añaden a esta clase son para habilitar e inhabilitar los elementos de la interfaz una vez el usuario haya solicitado la ejecución de un servicio.

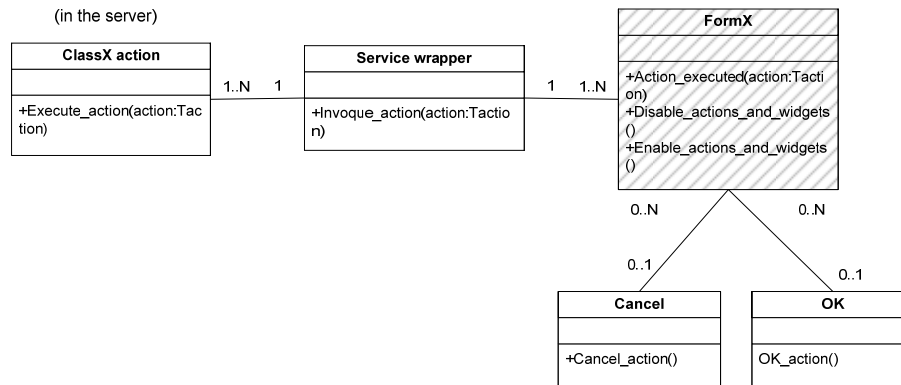


Figura 5.15 Diagrama de Clases para *Interaction Feedback*

La Figura 5.16 muestra en detalle la secuencia de ejecución de los nuevos métodos que se incorporan a la clase *FormX*. El usuario iniciará la ejecución del servicio pulsando sobre el botón *OK* de la ventana. Antes de solicitar la ejecución del servicio, la clase *Form X* inhabilitará sus botones y sus campos de entrada de datos con el método *Disable_actions_and_widgets*. Posteriormente, la clase *OK* iniciará la ejecución del servicio invocando al método *Invoke_action* de la clase *Service_wrapper*, la cual invocará mediante el método *Execute_action* a la clase del servidor que tenga la lógica asociada a la ejecución del servicio. Una vez finalizada la ejecución, la clase *Form X* invocará al método *Enable_action_and_widgets* para volver a habilitar los componentes de la ventana.

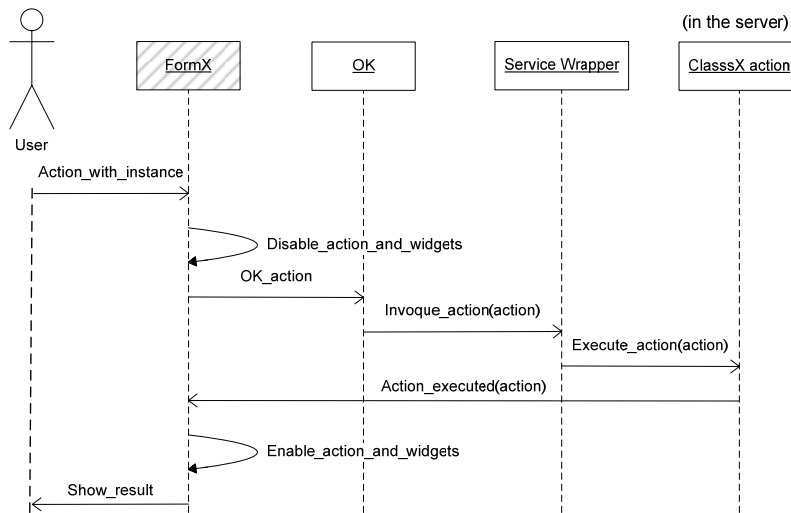


Figura 5.16 Diagrama de Secuencia para incorporar la forma de uso *Informar que la interacción está siendo atendida*

La Tabla 5.18 muestra un resumen de los cambios que incorpora WU_IF1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	<ul style="list-style-type: none"> - Inhabilita el botón desde el que se lanza el servicio mientras dure la ejecución de éste - Inhabilita los campos editables de la interfaz mientras dure la ejecución del servicio

Tabla 5.18 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Informar que la interacción está siendo atendida*

5.2.4.2 Resumen de los cambios que provoca Interaction Feedback

La Tabla 5.19 muestra de manera resumida la forma de uso del mecanismo de usabilidad *Interaction Feedback*, sus propiedades, los modelos conceptuales que se ven afectados, las nuevas primitivas conceptuales que se deberían incorporar y los cambios en el compilador de modelos.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Informar que la interacción está siendo atendida	Marcar un botón	Ninguno	Ninguno	La clase <i>Form X</i> resalta el botón pulsado durante su ejecución
	Inhabilitar componentes de la interfaz	Ninguno	Ninguno	La clase <i>Form X</i> inhabilita los componentes durante la ejecución
	Cambiar el puntero del ratón	Ninguno	Ninguno	Ninguno

Tabla 5.19 Resumen de los cambios para incorporar *Interaction Feedback*

5.3 Progress Feedback

El objetivo de este mecanismo de usabilidad es el de mostrar el tiempo restante para la finalización de un servicio lanzado por él usuario. Esta funcionalidad está pensada para aquellos servicios cuya ejecución requiere más de dos segundos [Tidwell, 2005] [Welie, 2000]. Además, para aquellos servicios cuya ejecución no sea crítica y su duración estimada sea de más de cinco segundos, el sistema debe dar la posibilidad al usuario de ejecutar otras tareas [Welie, 2000].

5.3.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *Progress Feedback* (Anexo I) se obtiene una única forma de uso, **Mostrar progreso de la ejecución (WU_PF1)**. La Tabla 5.20 muestra de forma resumida las preguntas de la guía de captura de requisitos de la que derivan

las formas de uso y los objetivos que se pretenden conseguir con cada de las formas de uso.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Cómo se informará al usuario sobre el progreso de las tareas que se ejecuten en el sistema?	Informar al usuario del tiempo restante para la finalización de la ejecución de un servicio	Mostrar progreso de la ejecución (WU_PF1)

Tabla 5.20 Derivación de las formas de uso de *Progress Feedback* a partir de la guía de captura de requisitos

La pregunta de la guía de captura de requisitos a partir de la cual se deriva esta forma de uso es, *¿Cómo se informará al usuario sobre el progreso de las tareas que se ejecuten en el sistema?* El objetivo de esta pregunta es el de determinar de entre todas las posibles alternativas, cómo se indicará al usuario el progreso de los servicios que ejecute.

Esta aplicación del mecanismo de usabilidad se utiliza, mientras se está ejecutando un servicio, para indicar cuánto progreso se ha hecho de esta ejecución. El progreso se puede indicar con la porción de tiempo (en porcentaje) que resta para completar la ejecución; mediante el tiempo real (en segundos o minutos) que resta para su finalización; mostrando los pasos que ya se han ejecutado y los que restan. Además de mostrar el progreso en la ejecución, el sistema debe proporcionar un mecanismo por si el usuario quiere abortar esta ejecución ante de que finalice en caso de que la ejecución se alargue mucho tiempo.

Por ejemplo, en el sistema de alquiler de coches, el servicio de reservar alquiler de coche requiere llevar a cabo varias comprobaciones (el cliente existe, el coche está disponible, los datos bancarios del cliente son correctos, etc.). Todas estas comprobaciones ralentizan la ejecución del servicio, y por tanto se debe mostrar un indicador de progreso indicando el tiempo restante para que el servicio de reservar alquiler de coche finalice.

5.3.2 Propiedades

A continuación se presentan las propiedades utilizadas para adaptar esta forma de uso a las exigencias de usabilidad del usuario.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué tareas es probable que necesiten más de dos segundos para su ejecución y cuales de ellas son críticas?	Indicar los servicios que mostrarán el progreso de su ejecución	Selección de servicios (P1_ WU_PF1)	Configurable
¿Cómo detener la ejecución de la tarea si el tiempo para que finalice es superior a diez segundos?	Permitir al usuario abortar la ejecución del servicio mientras visualiza su progreso	Abortar la ejecución (P2_ WU_PF1)	No configurable
¿Cuál es la porción de tarea completada? ¿Cuánto tiempo resta para finalizar? ¿Cómo se va a informar que la ejecución ha finalizado?	Determinar el aspecto visual del progreso de la ejecución del servicio	Visualización del progreso (P3_ WU_PF1)	Configurable

Tabla 5.21 Derivación de propiedades de la forma de uso *Mostrar progreso de la ejecución* a partir de la guía de captura de requisitos

Las **propiedades de Mostrar progreso de la ejecución (WU_PF1)** determinan los servicios que mostrarán el progreso de su ejecución y cómo se visualizará este progreso. La Tabla 5.21 muestra de forma global las preguntas de la guía de captura de requisitos de las que se derivan las propiedades y los objetivos que se pretenden alcanzar con cada propiedad.

Más detalladamente, las propiedades de *Mostrar progreso de la ejecución (WU_PF1)* son:

- **Selección de servicios (P1_ WU_PF1):** La pregunta de la guía de captura de requisitos de la cual se deriva esta propiedad es, *¿Qué tareas es probable que necesiten más de dos segundos para su ejecución y cuales de ellas son críticas?* El objetivo de esta pregunta es el de determinar de entre todos los servicios del sistema, cuáles pueden visualizar el progreso de su ejecución. Esta propiedad debe especificar cuáles son los servicios que indicarán el progreso de su ejecución al usuario. Según [Tidwell, 2005] [Welie, 2000], los servicios deberían ser aquellos que requieran más de dos segundos para su ejecución. Aunque, es difícil en fase de análisis determinar qué servicios requieren tanto tiempo, el analista debe seleccionar aquellos servicios cuya ejecución sea lenta, bien porque hacen varias consultas a la base de datos o bien porque tienen que ejecutar operaciones muy complejas. Esta propiedad es por tanto configurable.
- **Abortar la ejecución (P2_ WU_PF1):** La pregunta de la guía de la que se deriva esta propiedad es, *¿Cómo detener la ejecución de la tarea si el tiempo para que finalice es superior a diez segundos?* El objetivo de esta pregunta es proporcionar al usuario la capacidad de abortar la tarea si ésta se alarga mucho. Mientras se visualiza el progreso de ejecución de un servicio, el usuario debería poder cancelar este servicio a mitad ejecución. Esta propiedad es no configurable, ya que según Benson, debería estar en todos los sistemas que muestren el progreso de la ejecución de un servicio [Benson, 2002].
- **Visualización del progreso (P3_ WU_PF1):** Las preguntas de la guía de las cuales se deriva esta propiedad es, *¿Cuál es la porción de tarea completada? ¿Cuánto tiempo resta para finalizar? ¿Cómo se va a informar que la ejecución ha finalizado?* El objetivo de todas

estas preguntas es el de determinar cuál es la mejor opción para visualizar el progreso de ejecución de los servicios. Esta propiedad debe especificar los detalles de visualización del indicador de progreso. Debería determinar si el indicador se muestra de forma gráfica (barra de progreso) o de forma textual, es decir, con una lista en la que se van marcando las acciones que se van ejecutando. También debería determinar la posición del indicador de progreso (en una nueva ventana o en la ventana principal) y si el indicador se muestra en porcentaje de tiempo restante o en tiempo real (segundos, minutos). De entre todas estas posibles representaciones del progreso, el analista debe seleccionar una, por lo tanto esta propiedad es configurable.

A continuación se presenta un ejemplo de servicio al que se le ha aplicado la forma de uso, *Mostrar progreso de la ejecución (WU_PF1)*. Como servicio se ha seleccionado el de alquiler de coches. Por lo tanto, el analista ha dado el valor *alquiler de coches* a la propiedad *Selección de servicios (P1_WU_PF1)*. La propiedad *Abortar la ejecución (P2_WU_PF1)* al ser no configurable se añadiría al sistema de manera automática. Por último, la propiedad *Visualización del progreso (P3_WU_PF1)* podría tomar distintos valores, dependiendo de los gustos del usuario. Como ejemplo, en este estudio de laboratorio se han seleccionado algunas de estas combinaciones:

- Visualización gráfica del progreso (barra de progreso):
 - Visualización de forma gráfica en una ventana emergente (Figura 5.17 a)
 - Visualización de forma gráfica en una ventana emergente con indicador del tiempo restante relativo a la porción de tareas que implica la ejecución del servicio (Figura 5.17 b). En este ejemplo, se supone que el servicio alquiler de coche contiene cuatro tareas: comprobar que el cliente es correcto; comprobar los datos bancarios del cliente; comprobar que el coche seleccionado está disponible en las fechas indicadas; almacenar la reserva en el sistema
 - Visualización de forma gráfica en una ventana emergente con el tiempo real que resta para la finalización del servicio (Figura 5.18).

- Visualización de forma gráfica en la ventana principal mostrando el tiempo restante relativo a las tareas completadas que componen el servicio (Figura 5.19).

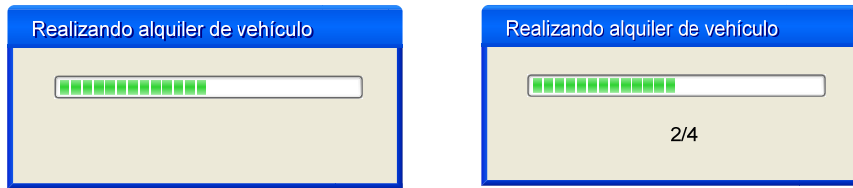


Figura 5.17 a) Barra de progreso sin mostrar tiempo restante textualmente.
b) Barra de progreso mostrando el tiempo restante relativo a las tareas completadas por el sistema

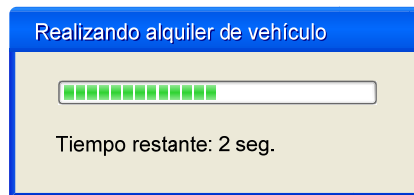


Figura 5.18 Barra de progreso mostrando el tiempo real restante para finalizar la ejecución

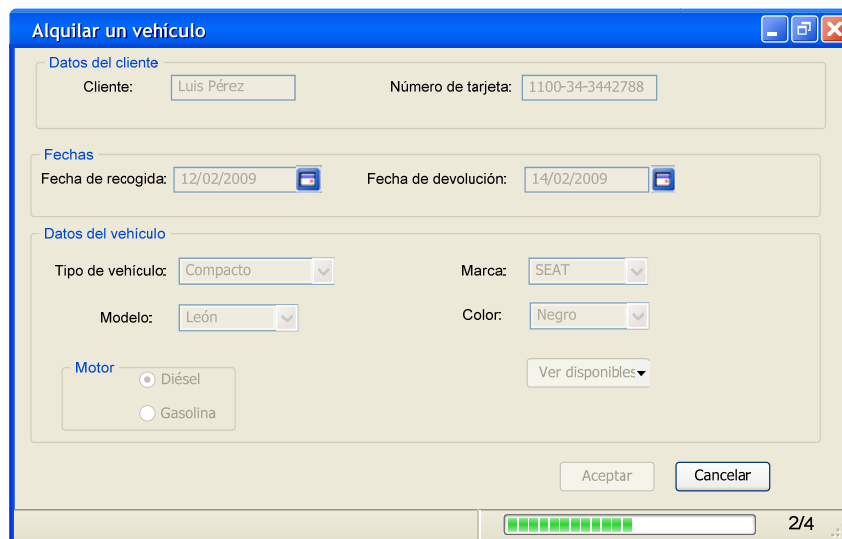


Figura 5.19 Barra de progreso dentro de la ventana principal

- Visualización textual del progreso:
 - Lista de tareas que componen el servicio, resaltando la tarea que se está ejecutando en el momento actual (Figura 5.20 a).
 - Mostrar la tarea que se está ejecutando en el momento actual y el tiempo relativo que resta para la finalización del servicio (Figura 5.20 b).

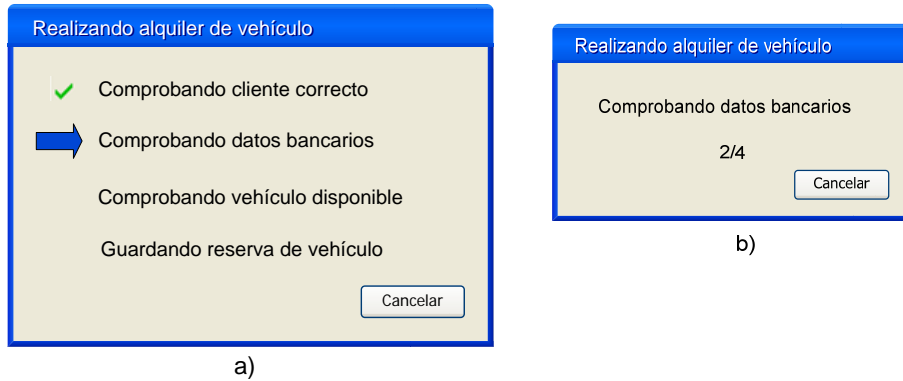


Figura 5.20 a) Progreso textual mostrando la lista de tareas que componen el servicio. b) Progreso textual mostrando sólo la tarea que se está ejecutando y el tiempo relativo que resta para la finalización de la ejecución

5.3.3 Modificación de los modelos conceptuales de OO-Method

Las aplicaciones generadas con OO-Method actualmente ya disponen de la funcionalidad de *Progress Feedback* para un caso particular. Cuando se seleccionan varias instancias de un objeto y se lanza una acción sobre ellas, aparece una barra de progreso indicando el tiempo que resta para que la acción finalice. Sin embargo, no se muestra ningún indicador de progreso cuando se ejecuta una transacción que engloba varios servicios. Las transacciones son un candidato ideal para usar la funcionalidad de este mecanismo porque engloban un conjunto de acciones complejas, y por tanto, el tiempo que tardan en ejecutarse puede ser considerable. El analista puede añadir la funcionalidad de un indicador de progreso a los dos tipos de transacciones existentes en OO-Method: transacciones globales y transacciones locales.

Por un lado, una transacción global agrupa un conjunto de acciones que pueden ser servicios y transacciones locales. En una transacción global se deben ejecutar todos los servicios que la componen y, si alguno de los servicios falla, toda la transacción global falla y ningún servicio de la transacción se ejecuta, es lo que se conoce como “ejecución todo o nada”. Además, las transacciones globales siguen el principio de atomicidad, es decir, no se pueden observar estados intermedios de la transacción, sino solo el resultado final. En las transacciones globales se debería mostrar un indicador de progreso para mostrar el tiempo restante para la finalización de su ejecución.

Por otro lado, el indicador de progreso también se puede aplicar a las transacciones locales, que al igual que las globales, están formadas por un conjunto de servicios y otras transacciones. También siguen el principio de atomicidad y de “ejecución todo o nada”. Se diferencian de las globales en que las locales solo afectan a una instancia de un objeto. Para este tipo de transacciones es también conveniente que el usuario vea en todo momento cuánto resta para la finalización de su ejecución.

Para el caso de ejecución de servicios simples sobre una única instancia no merece la pena aplicar este mecanismo, ya que su ejecución lleva muy poco tiempo y no se dispone del tiempo necesario para que el usuario vea el progreso de esta ejecución. Por lo tanto, la elección por parte del analista de los servicios van a llevar un indicador del progreso va a estar sólo entre el conjunto de transacciones locales y las transacciones globales.

En resumen, los cambios que incorpora a OO-Method el mecanismo de usabilidad *Progress Feedback* son:

- El analista debe tener la capacidad de seleccionar aquellas transacciones globales y locales que van a llevar un indicador del progreso.
- El sistema debe dar la funcionalidad al usuario de cancelar la ejecución de la transacción a mitad.

- El analista debe poder seleccionar cómo se visualiza el indicador de progreso: de forma gráfica, textual, con el tiempo real o relativo, en la ventana principal o en una ventana emergente.

La Tabla 5.22 muestra de forma esquemática los cambios que cada una de las propiedades de la forma de uso *Mostrar progreso de la ejecución (WU_PF1)* implica en OO-Method:

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Mostrar progreso de la ejecución (WU_PF1)	Selección de servicios (P1_WU_PF1)	Configurable	El analista debe seleccionar las transacciones globales y locales que mostrarán un indicador de progreso
	Abortar la ejecución (P2_WU_PF1)	No configurable	El sistema generado debe ser capaz de cancelar la transacción a mitad ejecución
	Visualización del progreso (P3_WU_PF1)	Configurable	El Analista debe decidir cómo visualizar el indicador de progreso

Tabla 5.22 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Progress Feedback*

A continuación se presentan las nuevas primitivas conceptuales que se deben incorporar en el modelo conceptual de OO-Method para soportar la forma de uso *Mostrar progreso de la ejecución*.

5.3.3.1 WU_PF1: Mostrar progreso de la ejecución

Las únicas propiedades configurables de WU_PF1 son la que selecciona las transacciones que van a llevar un indicador de progreso, llamada *Selección de servicios (P1_WU_PF1)* y la que configura las opciones de visualización de dicho indicador, llamada *Visualización del progreso (P3_WU_PF1)*. La selección de las transacciones que llevan indicador de progreso se realiza en el Modelo de Interacción Abstracto, mientras que el modelado de los aspectos visuales del indicador se realiza en el Modelo de Interacción Concreto.

Por un lado, en el **Modelo de Interacción Abstracto**, el analista debe disponer de una lista con todas las transacciones locales y globales que se han definido. Usando esta lista, el analista debe seleccionar aquellas transacciones en las que se deba incorporar un indicador de progreso. Por lo tanto, este modelo sólo incorpora una primitiva conceptual a través de la cual se realiza la selección de transacciones.

El valor por defecto que se muestra al analista en este modelo es que ninguna transacción tiene definido un indicador de progreso.

Por otro lado, en el **Modelo de Interacción Concreto** se modela la visualización del indicador de progreso. Este modelado se debe hacer para cada una de las transacciones tanto locales como globales y para cada uno de los servicios que mostrarán el indicador de progreso cuando se lance sobre un conjunto de instancias ese mismo servicio (selección múltiple de instancias). Para cada transacción o servicio seleccionado, en el Modelo de Interacción Concreto se decide entre mostrar el indicador de progreso de forma gráfica o textual. Las primitivas conceptuales para configurar estas dos opciones en el modelado conceptual son las siguientes:

- Mostrar indicador de progreso gráficamente (barra de progreso)
 - Ubicación de la barra de progreso:
 - En algún margen de la ventana principal
 - En una nueva ventana
 - Mostrar el porcentaje del tiempo que resta para su finalización
 - Mostrar tiempo restante en segundos para su finalización

- Desplazarse de izquierda a derecha o viceversa
- Mostrar indicador de progreso como una lista de todos los servicios que componen la transacción y resaltando aquellos cuya ejecución ya ha finalizado. Hay una Primitiva Conceptual por cada uno de las siguientes alternativas de representación de la barra de progreso:
 - Ubicación de la lista de servicios:
 - En la ventana principal
 - En una nueva ventana
 - Texto de la lista de servicios:
 - Fuente del texto
 - Tamaño
 - Color
 - Alineación
 - El listado de los servicios que forman la transacción local se podría hacer de una de estas tres formas:
 - Se muestran todos los servicios y se resalta el que se esté ejecutando en el mismo momento
 - Se muestra la lista de servicios conforme se vayan ejecutando
 - Se muestra solo el servicio que se ejecute sin mostrar el resto de servicios que forman la transacción local

El valor por defecto que este modelo proporciona al analista es un indicador de progreso gráfico (barra de progreso), que se muestra en una nueva ventana, y que se desplaza de izquierda a derecha.

La Tabla 5.23 muestra de forma resumida los cambios que incorpora esta forma de uso al modelado conceptual de OO-Method.

En la siguiente sección se presentan los cambios que hay que introducir en el compilador de Modelos para que el código generado refleje la configuración que el analista haya definido mediante estas primitivas.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Abstracto	Selección de las transacciones locales y globales que van a llevar un indicador de progreso

Modelo de Interacción Concreto	<ul style="list-style-type: none"> – Tipo de visualización: en texto o gráficamente. – Visualizar tiempo real o relativo – Formato y ubicación del indicador en la ventana del sistema
--------------------------------	---

Tabla 5.23 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Mostrar progreso de la ejecución*

5.3.4 Modificación del compilador de modelos

5.3.4.1 WU_PF1: Mostrar progreso de la ejecución

Los cambios introducidos en el compilador de modelos para incorporar las propiedades configurables y no configurables están representados en el Diagrama de Clases de la Figura 5.21.

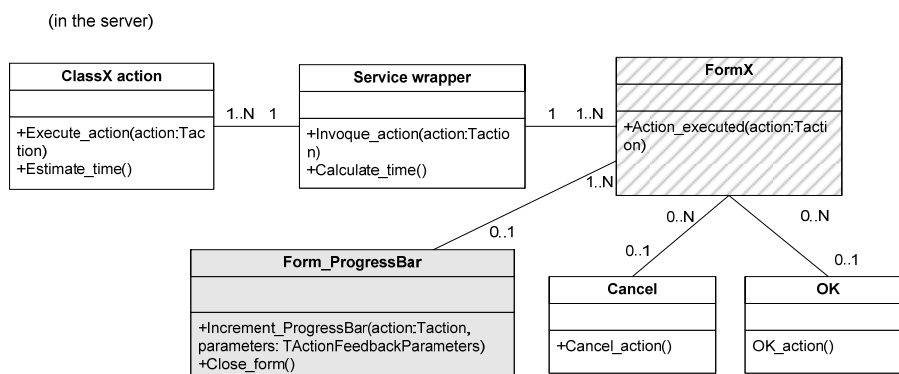


Figura 5.21 Diagrama de Clases para *Progress Feedback*

La clase *Form_ProgressBar* es la que almacena la apariencia visual que hubiera definido el analista para un servicio o transacción en concreto. Las instancias de esta clase van asociadas a instancias de la clase *FormX*, que es la que implementa las UI de Servicio. Se define esta relación entre clases porque el indicador de progreso se mostrará cada vez que el usuario lance la ejecución de una UI de Servicio con un indicador de progreso asociado.

El mecanismo de usabilidad *Progress Feedback* tiene tres aplicaciones distintas para OO-Method:

- La ejecución de un servicio sobre varias instancias seleccionadas.
- La ejecución de una transacción global
- La ejecución de una transacción local.

La primera aplicación no implica cambios importantes en el compilador porque ya está actualmente soportada. El único cambio es la inclusión del modelado del aspecto visual para el indicador de progreso. Este cambio no necesita ser explicado mediante Diagramas de Secuencia porque son sólo aspectos visuales y no afecta a los métodos para ejecutar los servicios.

La aplicación del patrón *Progress Feedback* a transacciones globales y locales requiere el uso de un Diagrama de Secuencia para explicar la secuencia en la llamada a los métodos en cada uno de los dos casos. Por un lado, las transacciones globales (Figura 5.22) se inician mediante el método *OK_action* cuando el usuario pulsa el botón OK en una UI de Servicio. En este momento el sistema calculará el número de instancias y de servicios implicados en la ejecución de la transacción para estimar el tiempo que le llevará la ejecución. Esta estimación se hará mediante los métodos *Calculate_time* y *Estimate_time*. Después, se entrará en un ciclo iterativo, donde por cada instancia se ejecutarán todos los servicios asociados a la transacción global. Mediante el método *Invoque_action* se invocará a *Execute_action* que ejecutará el servicio en el servidor. Cada vez que se ejecuten todos los servicios para cada una de las instancias, se incrementará el indicador de progreso en uno, usando para ello el método *Increment_ProgressBar*. Una vez ejecutados todos los servicios para todas las instancias implicadas, se cerrará la ventana del indicador de progreso usando el método *Close_Form*.

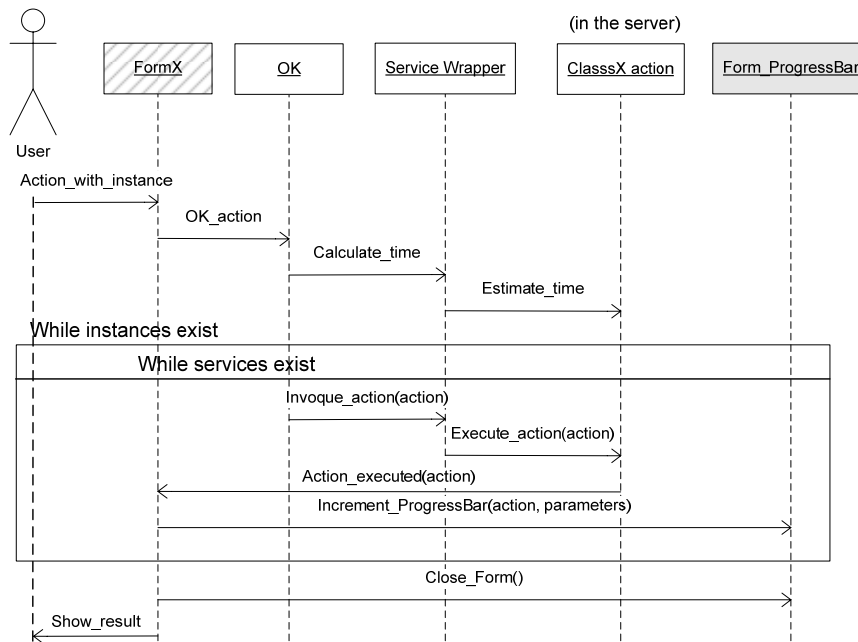


Figura 5.22 Diagrama de Secuencia para incorporar la forma de uso
Mostrar progreso de la ejecución en una transacción global

Por otro lado, la funcionalidad de las transacciones locales (Figura 5.23) se diferencia de la funcionalidad de las transacciones globales en que sólo contienen una iteración. En una misma iteración se recorren todos los servicios que componen la transacción local. El primer paso para aplicar el WU_PF1 es el de calcular el número de servicios que componen la transacción para estimar la duración de la ejecución. Como en el caso de transacciones globales, esta estimación se realizará mediante los métodos *Calculate_time* y *Estimate_time*. Una vez realizada la estimación, para cada acción se invocará al método *Invoke_action* de la clase *Service wrapper* que se encargará de hacer la petición de ejecución del servicio en la parte servidora. Esta petición se hará mediante el método *Execute_action* de la clase del servidor *ClassX action*. Al ejecutarse el servicio se informará a la clase *FormX* de que se ha terminado la ejecución. De esta forma, la clase *FormX* mediante el método *Increment_progress_bar* indicará que se debe incrementar el indicador de progreso mediante la clase *Form ProgressBar*, ya que uno de los servicios que componen la transacción local ha terminado. Una vez se hayan ejecutado todos los servicios que componen

la transacción local, se cerrará la ventana con la que se informa al usuario sobre su progreso.

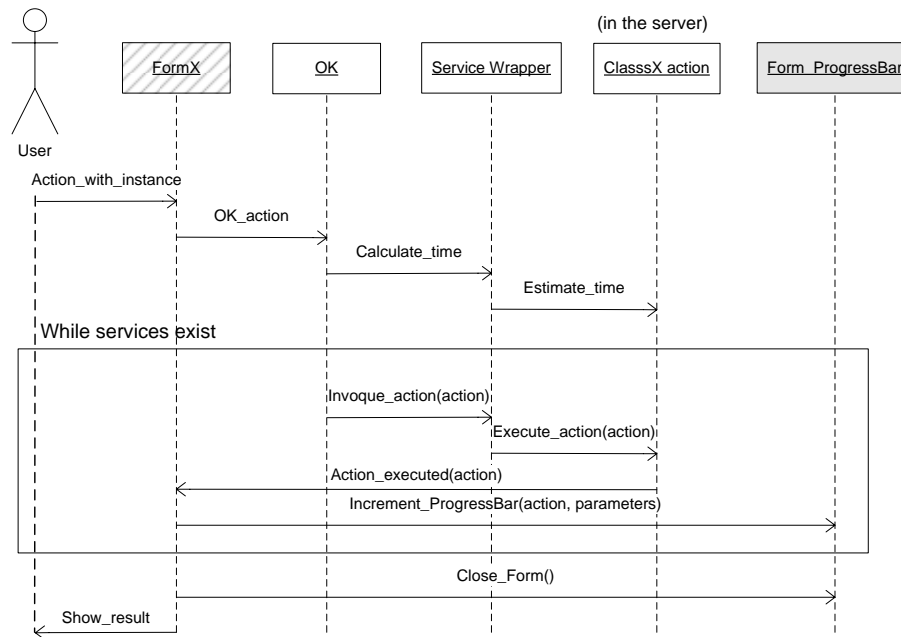


Figura 5.23 Diagrama de Secuencia para incorporar la forma de uso *Mostrar progreso de la ejecución* en una transacción local

La Tabla 5.24 muestra un resumen de los cambios que incorpora WU_PF1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	<ul style="list-style-type: none"> - Transacción global: para cada servicio que se ejecuta sobre cada instancia, esta clase informa a <i>Form_ProgressBar</i> que debe incrementar el indicador de progreso - Transacción local: para cada servicio que se ejecute, esta clase informa a <i>Form_ProgressBar</i> que debe incrementar el indicador de progreso

Clase Genérica	Cambios que Incorpora
Form_ProgressBar	<ul style="list-style-type: none"> – Incrementa el estado del indicador de progreso – Almacena las opciones de visualización de la barra de progreso

Tabla 5.24 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Mostrar progreso de la ejecución*

5.3.4.2 Resumen de los cambios que provoca Progress Feedback

La Tabla 5.25 muestra de manera resumida la forma de uso del mecanismo de usabilidad *Progress Feedback*, sus propiedades, los modelos conceptuales que se ven afectados, las nuevas primitivas conceptuales que se deben incorporar y los cambios en el compilador de modelos para generar el código.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Mostrar progreso de la ejecución	Selección de servicios	Modelo de Interacción Abstracto	Seleccionar las transacciones que mostrarán su progreso	La clase <i>FormX</i> indica si el servicio que se ejecuta en ella muestra progreso
	Abortar la ejecución	Ninguno	Ninguno	Ninguno (El cambio se explica en el mecanismo <i>Abort Operation</i>)

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
	Visualización del progreso	Modelo de Interacción Concreto	Seleccionar el tipo de visualización y su formato	<ul style="list-style-type: none"> – <i>Form X</i> indica cuándo incrementar la barra de progreso – <i>Form_ProgressBar</i> incrementa la barra y almacena opciones de visualización

Tabla 5.25 Resumen de los cambios para incorporar *Progress Feedback*

5.4 Warning

Este mecanismo de usabilidad tiene la funcionalidad de solicitar confirmación al usuario en caso de que el servicio solicitado tenga consecuencias irreversibles. De esta manera se evita que el usuario cometa errores. Además de solicitar esta confirmación, el sistema debería informar al usuario de las consecuencias que tiene la ejecución de la acción. Son varios los autores que han propuesto la funcionalidad de este mecanismo en forma de patrones de usabilidad, como Welie [Welie, 2000] con el patrón llamado *Escudo*, y Brighton con el patrón *Piénsalo dos veces* [Griffiths, 2002].

5.4.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *Warning* (Anexo I) se obtiene una única forma de uso **Mensaje de aviso (WU_W1)**.

La Tabla 5.26 muestra una visión global de la forma de uso, la pregunta de la guía de captura de requisitos a partir de la cual se ha derivado y el objetivo que se pretende conseguir con ella.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Qué tareas tienen serias consecuencias y por lo tanto requieren confirmación para el usuario?	Pedir confirmación para la ejecución de servicios con consecuencias irreversibles	Mensaje de aviso (WU_W1)

Tabla 5.26 Derivación de la forma de uso de *Warning* a partir de la guía de captura de requisitos

La pregunta de la guía para la captura de requisitos de la cual se deriva esta forma de uso es, *¿Qué tareas tienen consecuencias graves y por lo tanto requieren confirmación para el usuario?* El objetivo de esta pregunta es determinar qué tareas son irreversibles y pedir confirmación para su ejecución con el fin de evitar errores al usuario. Por lo tanto, la finalidad que se pretende alcanzar con esta pregunta es un subconjunto del heurístico de Nielsen llamado *Error prevention* [Nielsen, 1993].

Esta forma de uso tiene la funcionalidad de mostrar al usuario un mensaje de aviso que informa de las consecuencias de la ejecución de esa acción. Además, este mensaje debe dar la posibilidad al usuario de confirmar o rechazar la ejecución de la acción que ha provocado el mensaje.

Por ejemplo, en el sistema de alquiler de coches es raro que un cliente haga una reserva de alquiler para un periodo de más de un mes. Este caso es posible que sea un error en la introducción de las fechas. Por lo tanto, es recomendable que el sistema solicite confirmación al usuario antes de realizar una tarea que puede contener algún error en las fechas.

5.4.2 Propiedades

Una vez definida la forma de uso, el siguiente paso es el de definir las propiedades que adaptan esta forma de uso a los requisitos de usabilidad. Las **propiedades de Mensaje de aviso (WU_W1)** especifican los servicios que van a mostrar el mensaje de aviso y cómo se va a visualizar este mensaje.

Pregunta de la guía o patrón de usabilidad	Objetivo de la propiedad	Propiedad	Tipo
¿Qué tareas tienen consecuencias graves y por lo tanto requieren confirmación para el usuario?	Indicar los servicios que van a tener un mensaje de aviso definido	Selección de servicios (P1_WU_W1)	Configurable
Patrón <i>Warning</i> de Welie	Definir la condición que hará que se muestre el aviso	Condición (P2_WU_W1)	Configurable
– ¿Qué información se proporciona para confirmar cada una de las tareas? – Patrón <i>Warning</i> de Welie	Determinar el aspecto visual del mensaje de aviso	Visualización del mensaje (P3_WU_W1)	Configurable

Tabla 5.27 Derivación de propiedades de la forma de uso *Mensaje de aviso* a partir de la guía de captura de requisitos

La Tabla 5.27 muestra las preguntas de la guía de captura de requisitos o del patrón de usabilidad a partir de los cuales se derivan las propiedades.

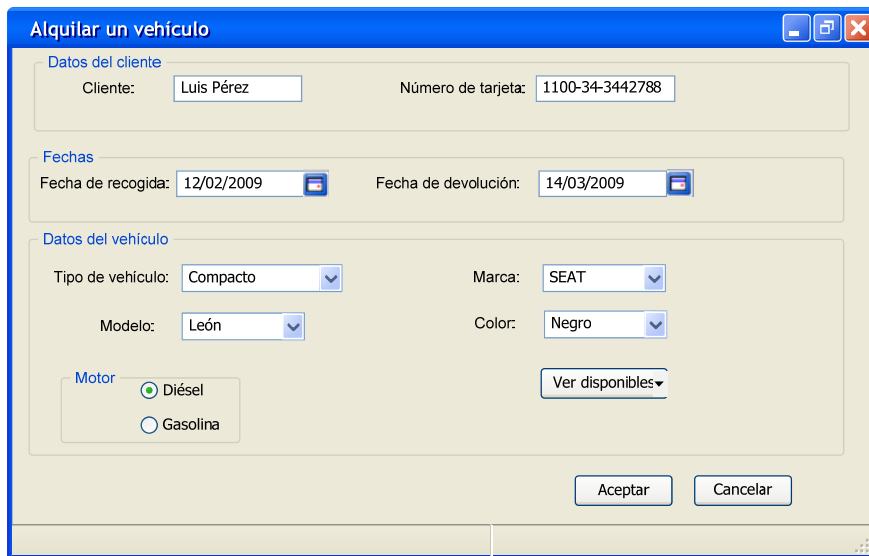
Además también incluye los objetivos de cada una de las propiedades definidas.

Específicamente, las propiedades que componen esta forma de uso son:

- **Selección de servicios (P1_WU_W1):** La pregunta de la guía de la que se extrae esta propiedad es *¿Qué tareas tienen consecuencias graves y por lo tanto requieren confirmación para el usuario?* El mensaje de aviso se debe definir para unos servicios específicos. Esta propiedad determina qué servicios mostrarán el aviso para confirmar su ejecución.
- **Condición (P2_WU_W1):** No hay ninguna pregunta en la guía de captura de requisitos que ayude al analista a definir la condición. En cambio, el patrón *Warning* de Welie sí que incluye la necesidad de definir la condición que lanza el mensaje de aviso [Welie, 2003]. Los mensajes de aviso se deben mostrar cuando cierta condición en base a la lógica de negocio se cumple. Esta propiedad es la que se encarga de definir dicha condición. Si el mensaje de aviso se ha de mostrar siempre, el analista debe poner esta condición a *cierto* para todos los casos.
- **Visualización del mensaje (P3_WU_W1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué información se proporciona para confirmar cada una de las tareas?* Welie en el patrón *Warning* también explica la necesidad de que el analista introduzca la explicación que se muestra al usuario en el mensaje de aviso [Welie, 2003]. Esta propiedad especifica cómo se visualizará el mensaje al usuario y el texto que se mostrará para pedir confirmación y prevenir al usuario de las consecuencias de la ejecución del servicio. Los mensajes de aviso deberían ser textuales, ya que la explicación del motivo de la alerta y la opción para confirmar o rechazar la ejecución de la acción no se puede presentar con un simple icono. Se pueden elegir aspectos de formato del texto y de la interfaz en la que se muestra este texto.

Como ejemplo para ilustrar los valores que pueden tomar estas propiedades se utiliza el servicio reservar alquiler de coche. Es un requisito que este servicio solicite confirmación al usuario antes de reservar un coche por un

periodo superior a un mes. Esta situación se produce pocas veces y es posible que el usuario se hubiera equivocado al introducir las fechas de recogida y de devolución. En el ejemplo de la Figura 5.24, al ser la diferencia entre fechas superior a un mes, cuando el usuario pulsara el botón de aceptar se mostraría el mensaje de aviso de la Figura 5.25. Para conseguir esta funcionalidad, la propiedad *Selección de servicios* ($P1_WU_W1$) tomaría el valor de reservar alquiler de coche. La propiedad *Condición* ($P2_WU_W1$) tomaría el valor de que la diferencia entre la fecha de recogida y de devolución fuera superior a un mes. La propiedad *Visualización del mensaje* ($P3_WU_W1$) tomaría el valor de un mensaje de alerta, con el texto y las características visuales de la Figura 5.25.



The screenshot shows a window titled "Alquilar un vehículo" with a blue header. It contains several sections: "Datos del cliente" with fields for "Cliente: Luis Pérez" and "Número de tarjeta: 1100-34-3442788"; "Fechas" with "Fecha de recogida: 12/02/2009" and "Fecha de devolución: 14/03/2009"; "Datos del vehículo" with dropdowns for "Tipo de vehículo: Compacto", "Marca: SEAT", "Modelo: León", and "Color: Negro"; and "Motor" with radio buttons for "Diésel" (selected) and "Gasolina". A "Ver disponibles" button is also present. At the bottom right are "Aceptar" and "Cancelar" buttons.

Figura 5.24 Ejemplo de ventana de Reservar alquiler de coche que muestra mensaje de aviso

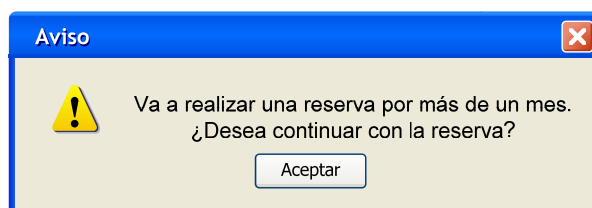


Figura 5.25 Ejemplo de mensaje de aviso para el servicio Reservar alquiler de coche

5.4.3 Modificación de los modelos conceptuales de OO-Method

OO-Method no proporciona soporte a la forma de uso **Mensaje de aviso (WU_W1)**, pero sí que da soporte a las precondiciones, cuya funcionalidad es parecida a la del mecanismo de usabilidad *Warning*. Mientras que en las precondiciones la condición definida por el analista se debe dar siempre para que se ejecute el servicio, en WU_W1, la condición sólo se utiliza para solicitar confirmación al usuario para la ejecución del servicio. Es decir, se puede asumir que un *Mensaje de aviso* es una precondición que se puede ejecutar bajo la responsabilidad del usuario. Esta semejanza conduce a que el modelado de *Mensaje de aviso* sea similar al de una Precondición.

Los cambios que hay que incorporar a OO-Method para que dé soporte a WU_W1 son:

- El analista debe poder seleccionar aquellos servicios sobre los que desea definir un mensaje de aviso porque su ejecución es irreversible.
- El analista debe poder definir la condición de la que va a depender la visualización del mensaje de aviso.
- El analista debe poder introducir el texto que se mostrará en el mensaje de aviso.
- El analista debe poder representar cómo desea visualizar el mensaje de aviso, es decir, detalles estéticos del mensaje de aviso.

Es importante remarcar, que aunque WU_W1 tiene tres propiedades, los cambios que afectan a OO-Method son cuatro. A la hora de incorporar la propiedad *Visualización del mensaje (P3_WU_W1)*, se ha separado el contenido del mensaje de texto de la forma en la que se muestra. Esta separación se ha hecho porque OO-Method dispone de dos vistas de su Modelo de Interacción: Abstracta y Concreta. La definición del mensaje de texto está contenida dentro de la vista Abstracta y la forma en la que se muestra el mensaje está contenida dentro de la vista Concreta.

La Tabla 5.28 muestra de forma resumida los cambios que se incorporan a OO-Method a partir de cada una de las propiedades de WU_W1.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Mensaje de aviso (WU_W1)	Selección de servicios (P1_WU_W1)	Configurable	El analista debe seleccionar los servicios con mensaje de aviso
	Condición (P2_WU_W1)	Configurable	El analista debe introducir la condición asociada al aviso
	Visualización del mensaje (P3_WU_W1)	Configurable	<ul style="list-style-type: none"> – El analista decide el texto del mensaje de aviso – El analista decide cómo visualizar el mensaje de aviso

Tabla 5.28 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Warning*

En la siguiente sección se detalla cómo afectan todos estos cambios al modelo conceptual de OO-Method.

5.4.3.1 WU_W1: Mensaje de aviso

Todas las propiedades de la única forma de uso de este patrón son configurables y por lo tanto es necesario añadir nuevas primitivas conceptuales para su modelado. Los modelos que se ven afectados por la

incorporación de este patrón son el Modelo de Objetos y el Modelo de Interacción Concreto.

Por un lado, las primitivas que se deben añadir al **Modelo de Objetos** son las encargadas de configurar la funcionalidad del mecanismo de usabilidad *Warning*. Se deben añadir tres nuevas Primitivas:

- El servicio sobre el que se aplica el patrón de usabilidad.
- La condición que se debería satisfacer para mostrar el mensaje de aviso.
- El texto que se mostrará al usuario cuando la condición se cumpla.

Por otro lado, las nuevas primitivas que se deben incorporar al **Modelo de Interacción Concreto** son las encargadas de representar cómo se visualizará el mensaje de texto una vez se cumpla la condición asociada al aviso. Se deben añadir primitivas para representar cada uno de los siguientes aspectos:

- Si la ventana es bloqueante (modal) o no. Por ventana bloqueante se entiende aquella ventana que no permite hacer ninguna otra operación con el sistema hasta que no se cierre. La ventana no bloqueante es aquella que bloquea la ventana que la ha lanzado, pero el usuario puede trabajar con el resto de ventanas del sistema. Por defecto la ventana sería no bloqueante.
- El tipo de ventana, es decir, si es una ventana de alerta, información, o error.
- Fuente del texto.
- Tamaño.
- Color.
- Alineación.

Los valores por defecto del Modelo de Interacción Concreto son una ventana bloqueante, de tipo alerta, fuente Arial, tamaño 10, color negro y alineación centrada.

La Tabla 5.29 muestra en resumen los cambios que incorpora esta forma de uso al modelado conceptual de OO-Method.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Objetos	<ul style="list-style-type: none"> – Servicio sobre el que se define el aviso – Condición de la que depende el visualizar el aviso – Texto que se mostrará en el aviso
Modelo de Interacción Concreto	<ul style="list-style-type: none"> – Si la ventana es modal – Tipo de ventana – Formato del texto del aviso

Tabla 5.29 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Mensaje de aviso*

5.4.4 Modificación del compilador de modelos

5.4.4.1 WU_W1: Mensaje de aviso

Todos los cambios que se llevan a cabo en el compilador de modelos para incorporar el patrón *Warning* se hacen con el objetivo de dar soporte a las primitivas descritas en la sección anterior. Al ser la forma de uso *Mensaje de aviso (WU_W1)* tan similar al concepto de precondición ya existente en OO-Method, el código que se genera para dar soporte a WU_W1 es muy similar al que ya se genera actualmente para implementar una precondición. La Figura 5.26 muestra el Diagrama de Clases con las clases que se ven afectadas por la incorporación de WU_W1. Las clases que se modifican son las clases que implementan las precondiciones actualmente. En estas clases se deben añadir los métodos necesarios para comprobar la condición (*Check_warning_message*) y ejecutarla cuando el usuario lo confirme (*User_order_execute_action* y *Execute_action*). La única clase que se crea desde cero, *Alert manager*, es la encargada de implementar la opciones de visualización del mensaje de aviso.

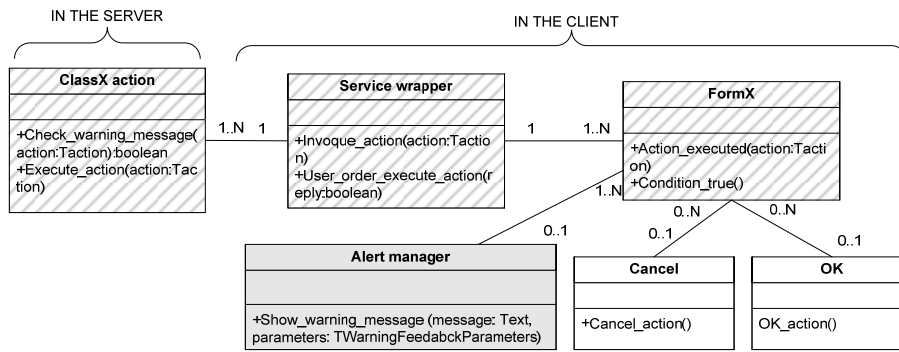


Figura 5.26 Diagrama de Clases para Warning

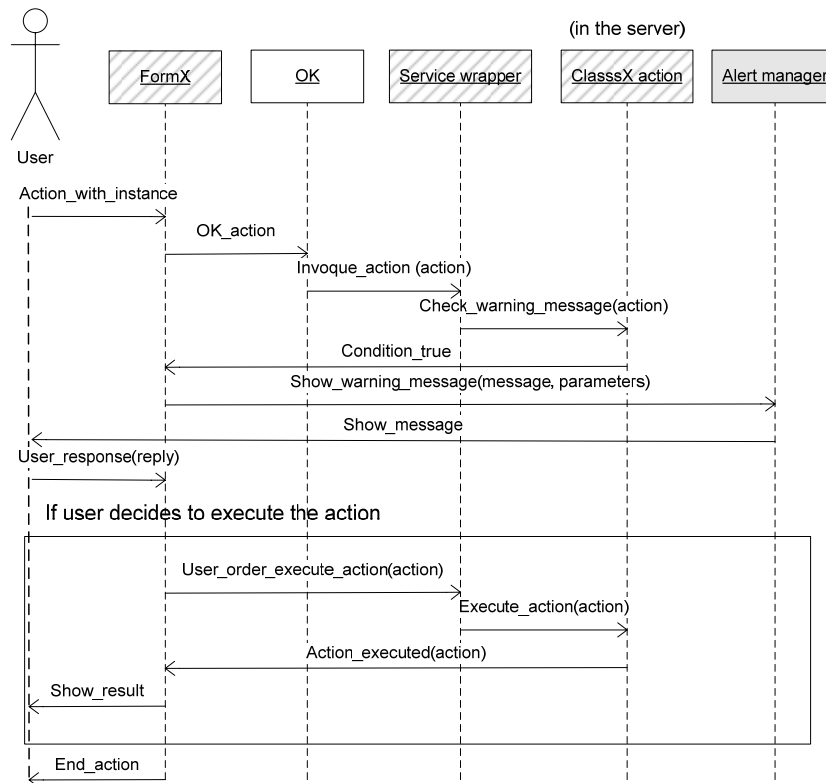


Figura 5.27 Diagrama de Secuencia para incorporar la forma de uso Mensaje de aviso

En la Figura 5.27 se puede apreciar el orden en la invocación de los métodos que utilizan el patrón Warning para su implementación (expuestos en la Figura 5.26). La secuencia de pasos mostrada representa el caso en

que la condición asociada al mensaje de aviso sea cierta y se deba mostrar el mensaje. Cuando un usuario quiera ejecutar un servicio, la clase *Service wrapper* recibirá la petición. Esta clase enviará la petición a la clase *ClassX action* implementada en el servidor. Esta clase se encargará de verificar si existe algún mensaje de aviso asociado con el servicio y, en este caso, comprobará si la condición se satisface. Para el caso en que la condición se satisfaga, la clase *Alert manager* preguntará al usuario si desea ejecutar el servicio o no. Si el usuario decidiera ejecutar el servicio a pesar de la advertencia, la clase *ClassX action* se encargará de ejecutarla. Si no deseara ejecutar el servicio, se terminará la ejecución del servicio.

La Tabla 5.30 muestra un resumen de los cambios que incorpora WU_W1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	<ul style="list-style-type: none"> – Invoca a la clase <i>Alert Manager</i> para que muestre el mensaje de aviso – Invoca a la clase <i>Service Wrapper</i> para que ejecute el servicio una vez el usuario haya confirmado su ejecución
Service Wrapper	Sólo debe ordenar la ejecución del servicio en la parte servidora cuando el usuario lo haya confirmado
Class X Action	Debe comprobar si la condición para que se muestre el mensaje de aviso se cumple o no
Alert Manager	<ul style="list-style-type: none"> – Esta nueva clase almacena las opciones de visualización del mensaje de aviso – Es la encargada de visualizar el aviso

Tabla 5.30 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Mensaje de aviso*

5.4.4.2 Resumen de los cambios que provoca Warning

La Tabla 5.31 muestra de manera resumida la forma de uso del mecanismo de usabilidad *Warning*, sus propiedades, los modelos conceptuales que se ven afectados, las nuevas primitivas conceptuales que se deben incorporar y los cambios en el compilador de modelos para generar el código.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Mensaje de aviso	Selección de servicios	Modelo de Objetos	Selección del servicio que mostrará el aviso	<ul style="list-style-type: none"> – <i>Service wrapper</i> ordena la ejecución del servicio tras la confirmación del usuario – <i>Form X</i> invoca a <i>Alert manager</i> para que muestre el aviso y a <i>Service wrapper</i> para la ejecución
	Condición	Modelo de Objetos	Definir la condición que se debe cumplir para mostrar el aviso	<i>Class X action</i> comprueba si la condición se cumple
	Visualización del mensaje	Modelo de Objetos	Definir el texto del mensaje de aviso	<i>Alert manager</i> almacena el texto

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
		Modelo de Interacción Concreto	Definir las opciones del formato de visualización	<i>Alert manager</i> almacena las opciones de visualización

Tabla 5.31 Resumen de los cambios para incorporar *Warning*

Capítulo 6

Incorporación de Wizard

Wizard es un FUF que tiene como objetivo ayudar paso a paso al usuario en la ejecución de tareas complejas. *Wizard* tiene un único mecanismo de usabilidad llamado *Step by Step*. Este capítulo tiene una única sección en la que se explica el resultado de aplicar el método MIMAT a dicho mecanismo de usabilidad.

6.1 Step by Step

Este mecanismo de usabilidad se utiliza para ayudar al usuario en tareas que son complejas y requieren la inserción de gran cantidad de datos. Está especialmente pensado para usuarios no expertos que deben realizar una tarea poco frecuente y compleja que se puede subdividir en varios pasos, en cada uno de los cuales el usuario debe interactuar con el sistema.

6.1.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *Step by Step* (Anexo I) se deriva una única forma de uso, **Definir un asistente (WU_SBS1)**.

La Tabla 6.1 muestra de forma global la pregunta a partir de la cual se ha derivado la forma de uso y el objetivo que se pretende conseguir con ella.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Qué tareas necesitan distintos pasos para ejecutarse?	Detectar aquellos servicios complejos que requieren de varios pasos para su ejecución	Definir un asistente (WU_SBS1)

Tabla 6.1 Derivación de la forma de uso de *Step by Step* a partir de la guía de captura de requisitos

La pregunta de la guía de captura de requisitos a partir de la cual se deriva esta forma de uso es *¿Qué tareas necesitan distintos pasos para ejecutarse?* El objetivo de esta pregunta es identificar aquellos servicios que, para ayudar al usuario en la ejecución, se dividen en varios pasos. Además de la pregunta de la guía, esta forma de uso también ha sido propuesta en el patrón de usabilidad de Welie llamado *Wizard* [Welie, 2000] y el patrón de usabilidad de Tidwell llamado *Step by Step* [Tidwell, 2005]. Esta aplicación del mecanismo de usabilidad hace que un servicio complejo se divida en varios pasos. En cada paso el sistema proporciona una serie de campos que el usuario debe rellenar, además de una breve descripción sobre el significado de estos campos. Los distintos pasos forman una especie de diagrama de flujo, donde la secuencia de los pasos puede depender de la información que el usuario haya introducido en pasos anteriores.

Por ejemplo, en el sistema de alquiler de coches se podría añadir un asistente para la acción de reservar un alquiler de coches, ya que es una acción compleja en la que el usuario debe introducir una gran cantidad de información.

6.1.2 Propiedades

Existen varias **propiedades para Definir un asistente (WU_SBS1)**. Estas propiedades especifican cómo se ejecutaría un servicio mediante un asistente, los pasos que van a componer el asistente y los aspectos visuales del asistente. La Tabla 6.2 da una visión global de las preguntas de las cuales se han extraído las propiedades y el objetivo que tiene asignado cada una de ellas.

Pregunta de la guía o patrón de usabilidad	Objetivo de la propiedad	Propiedad	Tipo
¿Qué tareas necesitan distintos pasos para ejecutarse?	Indicar el servicio sobre el que se define el asistente	Selección de servicio (P1_WU_SBS1)	Configurable
¿Cuántos pasos necesita cada tarea?	Determinar el número de pasos del asistente y qué acciones se hacen en cada paso	División en pasos (P2_WU_SBS1)	Configurable
¿Qué información necesita el usuario en cada paso?	Introducir una breve descripción de la acción que se hace en cada paso	Descripción de pasos (P3_WU_SBS1)	Configurable
Patrón <i>Wizard</i> de Welie	Determinar el orden de los pasos. Puede estar sujeto a condiciones	Flujo de ejecución de pasos (P4_WU_SBS1)	Configurable
¿Cuál es la mejor manera de presentar los pasos al usuario?	Determinar cómo se distribuyen los campos de cada paso	Disposición de los campos de los pasos (P5_WU_SBS1)	Configurable

Pregunta de la guía o patrón de usabilidad	Objetivo de la propiedad	Propiedad	Tipo
Patrón <i>Wizard</i> de Welie	Indicar el número de pasos restantes para finalizar la ejecución del asistente	Informar del número de pasos restantes (P6_WU_SBS1)	No configurable

Tabla 6.2 Derivación de propiedades de la forma de uso *Definir un asistente* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades de WU_SBS1 son las siguientes:

- **Selección de servicio (P1_WU_SBS1):** La pregunta de la guía de la que se deriva esta propiedad es, *¿Qué tareas necesitan distintos pasos para ejecutarse?* El objetivo de esta propiedad es el de seleccionar de entre toda la lista de servicios que forman el sistema, cuáles se ejecutarán a través del asistente.
- **División en pasos (P2_WU_SBS1):** Esta propiedad se deriva de la pregunta, *¿Cuántos pasos necesita cada tarea?* Esta propiedad tiene como objetivo el definir los distintos pasos que componen el asistente. Todos los argumentos que son necesarios para ejecutar el servicio se deben dividir entre los pasos que componen el asistente.
- **Descripción de pasos (P3_WU_SBS1):** Esta propiedad se obtiene de la pregunta de la guía, *¿Qué información necesita el usuario en cada paso?* Cada uno de los pasos puede llevar asociado un texto descriptivo que se mostrará al usuario. El objetivo de esta propiedad es el de añadir una descripción textual a cada uno de los pasos que componen el asistente. La descripción textual se introduce con el fin de ayudar al usuario a entender la información que se le solicita en cada paso.
- **Flujo de ejecución de pasos (P4_WU_SBS1):** No existe ninguna pregunta en la guía de captura de requisitos del mecanismo de

usabilidad referente a esta propiedad. Sin embargo, En el patrón de usabilidad *Wizard* de Welie [Welie, 2000] sí que se indica que el orden de los pasos del asistente puede depender de decisiones tomadas en pasos anteriores. Es decir, el orden de los pasos los debe definir el analista en base a ciertas condiciones. El objetivo de esta propiedad es el de indicar la secuencia de pasos en la ejecución del asistente. Dicha secuencia no tiene porqué ser la misma para todas las ejecuciones, puede depender de los valores introducidos en pasos anteriores. Por lo tanto, la secuencia entre dos pasos puede llevar asociada una condición.

- **Disposición de los campos de los pasos (P5_WU_SBS1):** Esta propiedad se obtiene de la pregunta de la guía, *¿cuál es la mejor manera de presentar los pasos al usuario?* Esta pregunta trata de determinar aspectos visuales de cada uno de los pasos del asistente. En esta propiedad define cómo se distribuyen en la interfaz los campos que componen un paso.
- **Informar del número de pasos restantes (P6_WU_SBS1):** Esta propiedad no se ha obtenido de la guía de captura de requisitos, sino del patrón *Wizard* de Welie. Según Welie, el usuario debe conocer los pasos que existen y los que ya se han completado. Es importante informar al usuario del número de paso en el que se encuentra y los que le restan para finalizar la ejecución del servicio. Esta propiedad debería estar en todos los asistentes según el patrón de usabilidad *Wizard* de Welie [Welie, 2000] y por tanto es una propiedad no configurable (aparecerá en todos los sistemas desarrollados).

Se va a utilizar el servicio reservar alquiler de coche del caso de caso de ilustración para mostrar la aplicación de estas propiedades. Este servicio requiere la introducción de gran cantidad de información por parte del usuario: el identificador del cliente, si el cliente es nuevo se debe dar de alta, los datos bancarios del cliente, las fechas en las que desea alquilar el coche y el tipo de coche que va a alquilar. Si todos estos datos los tuviera que rellenar el usuario en una única interfaz, esta interfaz sería similar a la de la Figura 5.4, pero añadiendo además campos para crear el cliente si éste no existiera en el sistema (Figura 5.1).

Al ser tantos los datos a proporcionar y estar la introducción de datos de algunos campos sujeta a una condición (si el cliente existe o no), lo mejor sería crear un asistente que guíe al usuario. De la Figura 6.1 a la Figura 6.7 se muestran las capturas de pantalla de una posible implementación del asistente. Cada figura representa cada uno de los pasos que compondrían el asistente: La Figura 6.1 representaría la identificación del cliente; la Figura 6.2, la Figura 6.3 y la Figura 6.4 se utilizarían para dar de alta el cliente en caso de que éste no exista; la Figura 6.5 permitiría introducir los datos bancarios del cliente a partir de los cuales se efectuará el pago, la Figura 6.6 sería la ventana desde la que se seleccionan las fechas del alquiler; finalmente, la Figura 6.7 sería la ventana desde la que se selecciona el coche.

Para este ejemplo, la propiedad *Selección de servicio (P1_WU_SBS1)* tomaría el valor del servicio reservar alquiler de coche. La propiedad *División en pasos (P2_WU_SBS1)* tomaría el valor de los siete pasos que componen el asistente. La propiedad *Descripción de pasos (P3_WU_SBS1)* tomaría el valor del texto descriptivo que se muestra en cada ventana del asistente. La propiedad *Flujo de ejecución de pasos (P4_WU_SBS1)* se configuraría con una condición: los pasos 2, 3 y 4 sólo se mostrarían si el usuario marca la casilla *Es un nuevo cliente* en el paso 1. El resto de pasos se mostrarían siempre de forma secuencial. La propiedad *Disposición de los campos de los pasos (P5_WU_SBS1)* tomaría la distribución que se muestra en las capturas de pantalla. Finalmente, la propiedad *Informar del número de pasos restantes (P6_WU_SBS1)* mostraría en todo momento el paso en el que se encuentra el usuario y los pasos que resten para la finalización. Es importante resaltar, que el número de pasos que resten para la finalización es el número máximo de pasos, pero puede que no se muestren todos los pasos restantes (dependiendo del flujo entre pasos). En el ejemplo, aunque haya 7 pasos, los pasos a visualizar serían 4 si el usuario ya estuviera registrado.

The screenshot shows a window titled "Alquilar un vehículo 1/7" with a close button in the top right. The main area is titled "Introduzca la identificación del cliente". On the left, there is a dark blue vertical bar with a white icon of a car and a key. The form contains three input fields: "Nombre:" (empty), "Apellidos:" (empty), and "DNI:" (empty). Below these is a checked checkbox labeled "Es un nuevo cliente". At the bottom, there are three buttons: "< Atrás", "Siguiete >", and "Cancelar".

Figura 6.1 Paso1: Identificar cliente

The screenshot shows a window titled "Alquilar un vehículo 2/7" with a close button in the top right. The main area is titled "Datos del nuevo cliente". On the left, there is a dark blue vertical bar with a white icon of a car and a key. The form contains four input fields: "Nombre:" with the value "Luis", "Apellidos:" with the value "Pérez Martínez", "DNI:" with the value "335687902L", and "Fecha de nacimiento:" with the value "7/11/1981" and a calendar icon. At the bottom, there are three buttons: "< Atrás", "Siguiete >", and "Cancelar".

Figura 6.2 Paso 2: Introducir datos del nuevo cliente

Alquilar un vehículo 3/7

Dirección del nuevo cliente

Domicilio: Colón Núm: 70 Pta: 3

Provincia: Valencia

Localidad: Alboraya

Código postal: 46120

< Atrás Siguiente > Cancelar

Figura 6.3 Paso3: Introducir dirección del nuevo cliente

Alquilar un vehículo 4/7

Otros datos del nuevo cliente

Posee Carné de conducir

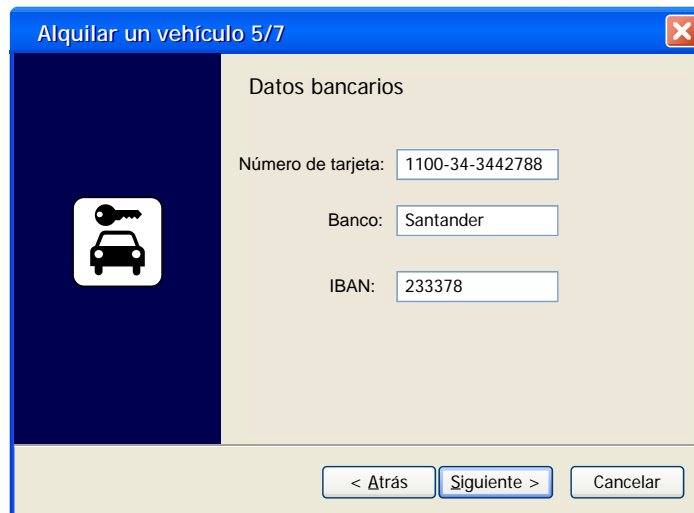
Estado civil

Soltero

Casado

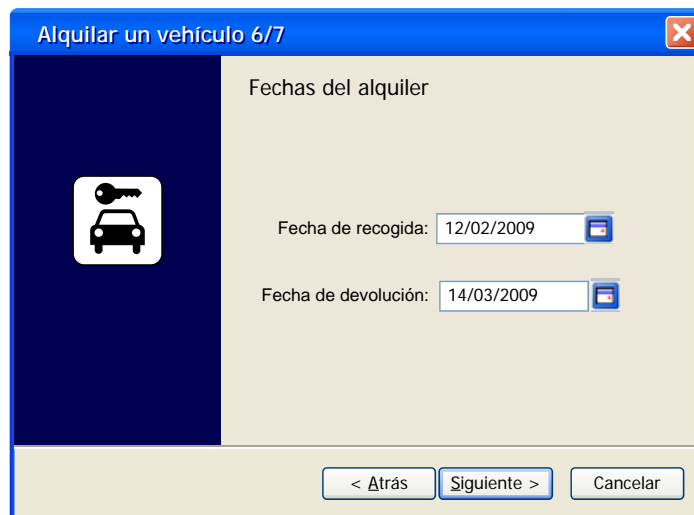
< Atrás Siguiente > Cancelar

Figura 6.4 Paso 4: Introducir otros datos del cliente



The screenshot shows a wizard window titled "Alquilar un vehículo 5/7". On the left is a dark blue sidebar with a white icon of a car and a key. The main area is titled "Datos bancarios" and contains three input fields: "Número de tarjeta:" with the value "1100-34-3442788", "Banco:" with the value "Santander", and "IBAN:" with the value "233378". At the bottom are three buttons: "< Atrás", "Siguiete >", and "Cancelar".

Figura 6.5 Paso 5: Introducir datos bancarios



The screenshot shows a wizard window titled "Alquilar un vehículo 6/7". On the left is a dark blue sidebar with a white icon of a car and a key. The main area is titled "Fechas del alquiler" and contains two date input fields: "Fecha de recogida:" with the value "12/02/2009" and "Fecha de devolución:" with the value "14/03/2009". Each date field has a small calendar icon to its right. At the bottom are three buttons: "< Atrás", "Siguiete >", and "Cancelar".

Figura 6.6 Paso 6: Introducir fechas del alquiler



Figura 6.7 Paso 7: Seleccionar coche

La siguiente sección muestra cómo las propiedades configurables del mecanismo *Step by Step* afectan al modelo conceptual de OO-Method.

6.1.3 Modificación de los modelos conceptuales de OO-Method

OO-Method no dispone en la actualidad de ninguna primitiva para modelar un asistente. Para ejecutar servicios que llevan asociados una gran cantidad de argumentos, OO-Method propone la utilización del Patrón Elemental de Agrupación. De esta forma se agrupan en distintas pestañas aquellos argumentos que tienen alguna relación en su significado. Sin embargo, esto tiene algunos inconvenientes que se solucionarían con un asistente: las pestañas son siempre las mismas (no hay posibilidad de flujos alternativos); el orden para rellenar los campos de las pestañas no se puede definir; no se puede proporcionar una descripción al usuario de cuál es el significado de los campos en cada una de las pestañas.

Por tanto, los cambios que se deben incorporar a OO-Method para soportar la forma de uso **Definir un asistente (WU_SBS1)** son:

- El analista debe poder indicar los servicios que se ejecutarán mediante un asistente.

- El analista debe poder decidir qué argumentos del servicio se rellenan en cada paso.
- El analista debe poder mostrar al usuario una breve descripción de las tareas que debe hacer en cada paso.
- El analista debe poder establecer el orden de los pasos en base a condiciones basadas en información almacenada en el sistema o a los pasos anteriores del asistente.
- El analista debe poder ubicar los argumentos del servicio para cada paso en el lugar más conveniente para el usuario.
- El sistema generado debe indicar, sin la intervención del analista, el paso en el que se encuentra el usuario y los pasos restantes para finalizar.

El Patrón Elemental del Modelo de Interacción Abstracto llamado *Dependencia* está relacionado con la forma de uso WU_SBS1. Es decir, se pueden establecer dependencias entre argumentos del asistente. Tal y como se explica en [Molina, 2003], las reglas de dependencia se describen con la nomenclatura evento-condición-acción. Un evento desencadena la ejecución de una acción si cierta condición es cierta. Estas relaciones pueden ser entre argumentos de un mismo paso o entre argumentos de diferentes pasos. Si la acción de la regla de dependencia afecta a argumentos del mismo paso que ocasiona el evento o a argumentos de futuros pasos, se ejecutará la regla y el usuario podrá visualizar el valor del argumento que recibe la acción de la regla (bien en el paso actual o en siguientes pasos). En caso de que el argumento que recibe la acción de la regla esté en algún paso anterior al paso que produce el evento de la regla, el usuario no podrá visualizar el resultado de la acción de dicha regla. Por lo tanto, en este caso se debe informar al usuario, mediante un mensaje textual, de la modificación que se ha hecho en los argumentos de pasos anteriores para que sea consciente de las modificaciones.

La Tabla 6.3 muestra la relación entre los cambios que se deben incorporar a OO-Method y las propiedades de *Definir un asistente (WU_SBS1)* de las que se derivan dichos cambios. En la siguiente sección se muestran en detalle las nuevas primitivas derivadas de las propiedades configurables.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Definir un asistente (WU_SBS1)	Selección de servicio (P1_WU_SBS1)	Configurable	El analista debe seleccionar el servicio que se ejecuta con un asistente
	División en pasos (P2_WU_SBS1)	Configurable	El analista debe dividir la inserción de argumentos del servicio en varios pasos
	Descripción de pasos (P3_WU_SBS1)	Configurable	El analista puede introducir una descripción para cada paso
	Flujo de ejecución de pasos (P4_WU_SBS1)	Configurable	El analista debe definir la secuencia de pasos. Puede estar sujeta a condiciones
	Disposición de los campos de los pasos (P5_WU_SBS1)	Configurable	El analista debe decidir cómo se distribuyen los argumentos en la interfaz
	Informar del número de pasos restantes (P6_WU_SBS1)	No configurable	El sistema debe informar al usuario de los pasos que restan para terminar el asistente

Tabla 6.3 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Step by Step*

6.1.3.1 WU_SBS1: Definir un asistente

A partir de la forma de uso llamada *Definir un asistente (WU_SBS1)* y sus propiedades, esta sección explica cómo modelar estas propiedades configurables dentro del modelo conceptual de OO-Method. Tal y como se ha comentado anteriormente, no hay ninguna propiedad de WU_SBS1 que esté actualmente soportada por OO-Method, por lo tanto, se deben definir nuevas primitivas conceptuales que soporten las propiedades configurables. Los modelos de OO-Method que se verían afectados por estas propiedades son el Modelo de Interacción Abstracto y el Modelo de Interacción Concreto.

Por un lado, en el **Modelo de Interacción Abstracto** se representa de forma abstracta que un servicio del sistema se va a ejecutar por medio de un asistente. Para ello se debe añadir la Unidad de Interacción Asistente a las Unidades de Interacción ya existentes (Servicio, Población, Instancia y Maestro/Detalle). Las nuevas primitivas conceptuales que se añaden para modelar la nueva Unidad de Interacción Asistente son:

- El servicio sobre el que se define el asistente.
- División de argumentos del servicio en pasos.
- Cada paso puede llevar una cadena de texto descriptivo.
- Definición de la secuencia de pasos. Esta secuencia se definiría mediante expresiones regulares en las cuales se pueden expresar condiciones en base a los siguientes parámetros:
 - Valor de atributos de una clase.
 - Argumentos del servicio a los que ya se les ha asignado un valor en pasos anteriores o en el propio paso.
 - Funciones de usuario.
 - Funciones estándar.
 - Operadores.

La herramienta de modelado de OO-Method facilita la definición de estas expresiones regulares mediante asistentes.

Estas primitivas del Modelo de Interacción Abstracto no presentan valores por defecto porque son totalmente dependientes de un sistema y no se puede predecir un posible valor para ellas.

Por otro lado, el **Modelo de Interacción Concreto** se debe enriquecer con nuevas primitivas para representar cómo se visualizan los distintos pasos del asistente. La única primitiva conceptual que añade WU_SBS1 es la que determina la posición de los campos de entrada de cada paso. Mediante esta primitiva, el analista puede especificar en qué posición de la ventana se ubicará cada campo de entrada para cada paso del asistente.

Esta nueva primitiva conceptual se muestra al analista con un valor por defecto en el que en cada línea de la interfaz se muestra un único campo de entrada de datos.

La Tabla 6.4 muestra de forma resumida los cambios que incorpora WU_SBS1 al modelado conceptual de OO-Method:

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Abstracto	<ul style="list-style-type: none"> – Servicio sobre el que se define el asistente – Clasifica los argumentos entre los pasos del asistente – Texto descriptivo que se mostrará por cada paso – Orden de visualización de los pasos (puede estar basado en condiciones)
Modelo de Interacción Concreto	La posición de los argumentos de entrada en la ventana

Tabla 6.4 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Definir un asistente*

La propiedad *Informar del número de pasos restantes (P6_WU_SBS1)* no tiene ninguna primitiva conceptual porque es una propiedad que interesa que aparezca en todos los sistemas, ya que ayuda a mejorar la usabilidad. Por ello, esta propiedad la incorpora a la aplicación el compilador de modelos sin que el analista tomara ninguna decisión. A continuación se presentan los cambios que esta propiedad y las nuevas primitivas conceptuales introducen en el compilador de modelos.

6.1.4 Modificación del compilador de modelos

6.1.4.1 WU_SBS1: Definir un asistente

Las propiedades de la forma de uso *Definir un asistente (WU_SBS1)* implican cambios en el compilador de modelos. El código generado debe reflejar la configuración de las propiedades expresadas mediante primitivas conceptuales, así como la propiedad que no tiene representación conceptual (*Informar del número de pasos restantes*). El Diagrama de Clases que representa de forma abstracta los cambios que introduce el mecanismo de usabilidad *Step by Step* sobre el código generado se reflejan en la Figura 6.8.

La única clase que se añadiría a las ya generadas actualmente es la clase llamada *StepX*. Esta clase representa de forma abstracta cada uno de los pasos que componen el asistente. Existe una instancia de esta clase para cada uno de los pasos del asistente. Los atributos que se deben incluir en esta nueva clase son:

- *Id* es el identificador del paso.
- *Name* es el nombre del paso que se corresponde con el título que se mostrará al usuario en la interfaz que lo represente.
- *Description* contiene el mensaje textual que sirve de ayuda al usuario en cada paso.
- *Arguments* es una lista con todos los argumentos que se deben rellenar en el paso.
- *Remaining_Steps* se utiliza para contabilizar el resto de pasos para finalizar la ejecución del servicio.
- *Condition* es la lista de condiciones que haya definido el analista para obtener cuál será el próximo paso.

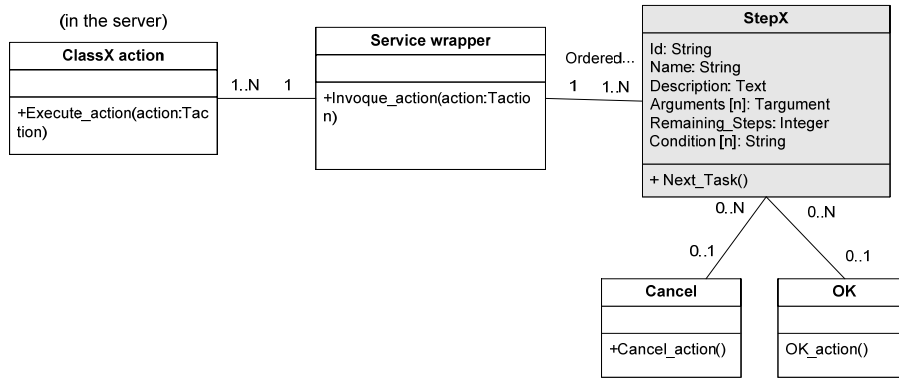


Figura 6.8 Diagrama de Clases para Step by Step

A partir de las clases presentadas anteriormente se puede construir un Diagrama de Secuencia (Figura 6.9) para la forma de uso *Definir un asistente (WU_SBS1)*. La clase *Step1*, una vez el usuario haya rellenado todos sus campos (usando el evento *Fill_in_arguments*), invocará al siguiente paso mediante el evento *Next_step*. La clase *Step X* calculará el siguiente paso y hará la navegación hacia él. Una vez rellenados todos los campos de entrada de todos los pasos (en el último paso), se invocará a la clase *ClassX_action* donde se ejecutará el servicio asociado al asistente.

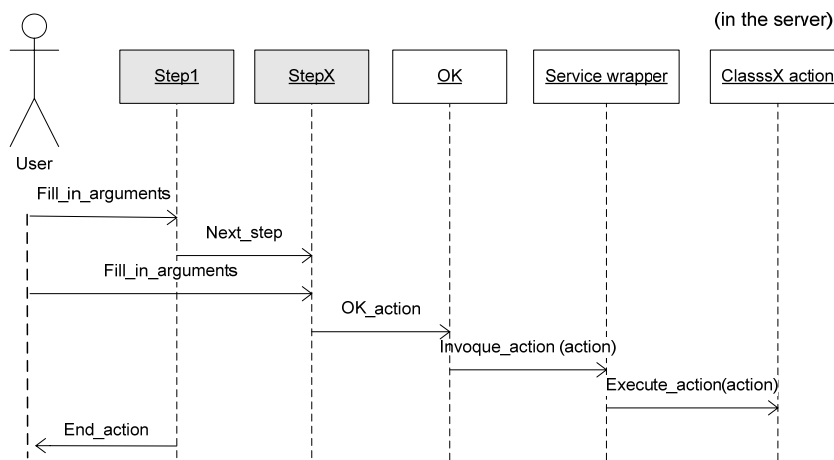


Figura 6.9 Diagrama de Secuencia para incorporar la forma de uso *Definir un asistente*

La Tabla 6.5 muestra un resumen de los cambios que incorpora WU_SBS1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Step X	<ul style="list-style-type: none"> – Engloba un conjunto de argumentos del servicio que hay detrás del asistente – Es capaz de averiguar mediante una fórmula lógica cuál es el siguiente paso – Almacena una breve descripción del paso – Almacena la posición de los argumentos que se mostrarán en este paso – Almacena el número de pasos totales del asistente – Invoca a la clase <i>Service Wrapper</i> una vez que se hayan rellenado los argumentos de todos los pasos del asistente

Tabla 6.5 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Definir un asistente*

6.1.4.2 Resumen de los cambios que provoca Step by Step

La Tabla 6.6 muestra de manera resumida la forma de uso del mecanismo de usabilidad *Step by Step*, las propiedades en las que se divide, los modelos conceptuales afectados, las nuevas primitivas conceptuales que se deben añadir y los cambios en el compilador de modelos para generar todo el código que implemente el mecanismo de usabilidad.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Definir un asistente	Selección de servicio	Modelo de Interacción Abstracto	Selección del servicio que se ejecutará con asistente	– <i>Step X</i> almacena el servicio y los argumentos de la ejecución

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
				– Una vez completados todos los pasos se invoca el servicio mediante <i>Service wrapper</i>
	División en pasos	Modelo de Interacción Abstracto	Definir los pasos del asistente	Cada clase <i>Step X</i> es un paso del asistente
	Descripción de pasos	Modelo de Interacción Abstracto	Definir una breve descripción en cada paso del asistente	<i>Step X</i> almacena la descripción
	Flujo de ejecución de pasos	Modelo de Interacción Abstracto	Definir el flujo de pasos del asistente en base a condiciones	<i>Step X</i> calcula el siguiente paso del asistente en base a una condición almacenada
	Disposición de los campos de los pasos	Modelo de Interacción Concreto	Definir dónde van ubicados los elementos de la interfaz de cada paso	<i>Step X</i> almacena la posición de los argumentos en cada paso
	Informar del número de pasos restantes	Ninguno	Ninguno	<i>Step X</i> almacena el número de pasos totales del asistente

Tabla 6.6 Resumen de los cambios para incorporar *Step by Step*

Capítulo 7

Incorporación de User Input Error Prevention

User Input Error Prevention es un FUF que tiene como objetivo evitar que el usuario cometa errores en el proceso de introducción de datos. *Input Error Prevention* tiene un único mecanismo de usabilidad llamado *Structured Text Entry*. Este capítulo tiene una única sección en la que se explica el resultado de aplicar el método MIMAT al mecanismo de usabilidad *Structured Text Entry*.

7.1 Structured Text Entry

Este mecanismo de usabilidad se utiliza para especificar elementos de la interfaz que facilitan la entrada de datos con un formato específico. El objetivo de este mecanismo es el mismo que el del patrón de usabilidad *Mostrar el formato requerido* de Brighton [Griffiths, 2002], y el del patrón *Formato inequívoco* de Welie [Welie, 2000].

7.1.1 Formas de uso

Aplicando el método de incorporación de los mecanismos de usabilidad al mecanismo de usabilidad *Structured Text Entry* (Anexo I), aparecen las siguientes formas de uso:

1. WU_STE1: Especificar el tipo de visualización del campo de entrada.
2. WU_STE2: Definición de máscaras.

3. WU_STE3: Valores por defecto.

La Tabla 7.1 muestra una visión global de las formas de uso, las preguntas de la guía de captura de requisitos a partir de las cuales se han derivado y los objetivos que se pretenden conseguir con cada una de las formas de uso.

Pregunta de la Guía y Patrones de Usabilidad	Objetivo de la forma de uso	Forma de uso
¿Cuál es el formato de los argumentos de entrada para el usuario?	Especificar el formato del campo de entrada para ayudar al usuario	Especificar el tipo de visualización del campo de entrada
¿Cómo guiar al usuario para que introduzca los datos en el formato requerido?	Evitar que el usuario introduzca datos con un formato no válido	Definición de máscaras
¿Cómo guiar al usuario para que introduzca los datos en el formato requerido? y el patrón de usabilidad de Brighton, Mostrar el formato requerido.	Guiar al usuario para que sepa cuál es el formato que debe utilizar en la entrada de datos	Valores por defecto

Tabla 7.1 Derivación de las formas de uso de *Structured Text Entry* a partir de la guía de captura de requisitos

La primera de las formas de uso, **Especificar el tipo de visualización del campo de entrada (WU_STE1)**, se deriva de la pregunta de la guía de captura de requisitos *¿Cuál es el formato de los argumentos de entrada para el usuario?* El objetivo de esta forma de uso es el de especificar el

formato del campo de entrada para ayudar al usuario en la introducción de datos.

Por ejemplo, en el sistema de alquiler de coches, dentro de la interfaz para alta de cliente, el argumento *posee carné de conducir* que es de tipo booleano (sí, o no) se puede representar de muy distintas formas: mediante un checkbox, mediante un radiobutton o mediante un simple campo editable donde el usuario tenga la libertad de introducir lo que desee. Para este ejemplo en concreto, sería más conveniente seleccionar una visualización en checkbox o en radiobutton, ya que en un campo editable el usuario podría introducir valores con un formato no permitido.

La segunda forma de uso, **Definición de máscaras (WU_STE2)**, se obtiene de la pregunta de la guía de captura de requisitos *¿Cómo guiar al usuario para que introduzca los datos en el formato requerido?* El objetivo de esta forma de uso es impedir que el usuario pueda introducir datos en un formato distinto al permitido. Las máscaras de edición es un mecanismo utilizado a nivel de implementación para dotar de un formato específico a un campo de introducción de datos. Llevando las máscaras a un mayor nivel de abstracción, el analista puede, mediante una fórmula regular, especificar el único formato que un campo editable puede proporcionar al sistema. Por ejemplo, en el servicio alta de cliente del sistema de alquiler de coches, el campo de entrada *Fecha de nacimiento* podría incluir la definición de una máscara que sólo permitiera la entrada de fechas con el formato *dd/mm/aaaa*.

Por último, la forma de uso **Valores por defecto (WU_STE3)** se deriva también de la pregunta de la guía de captura de requisitos *¿Cómo guiar al usuario para que introduzca los datos en el formato requerido?* y del patrón de usabilidad de Brighton [Griffiths, 2002] *Mostrar el formato requerido*. Aunque la pregunta de la guía de la que se deriva WU_STE2 y WU_STE3 es la misma, el objetivo de estas formas de uso es distinto. Mientras que WU_STE2 tiene como objetivo el impedir que el usuario introduzca datos en un formato incorrecto, WU_STE3 tiene como objetivo el mostrar mediante valores por defecto cuál debería ser el formato de los datos. Es decir, la aplicación de WU_STE3 no impide que el usuario introduzca los datos en un formato incorrecto.

Por ejemplo, en el servicio alta de cliente del sistema de alquiler de coches, el campo *Fecha de nacimiento* podría mostrar un valor por defecto con la fecha 10/12/1990. De esta forma el usuario sabe el formato que debe utilizar y puede cambiar el valor del campo para adaptarlo a su fecha de nacimiento.

7.1.2 Propiedades

Cada una de las formas de uso del mecanismo de usabilidad *Structured Text Entry* tiene varias propiedades con las que adaptar su funcionalidad. En cuanto a la forma de uso **Especificar el tipo de visualización del campo de entrada (WU_STE1)**, tiene una única propiedad mediante la cual determinar el formato del campo de introducción de datos. La Tabla 7.2 presenta la pregunta de la guía de captura de requisitos de la que se ha derivado la propiedad y sus objetivos.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Cuál es el formato de los campos de entrada?	Determinar el formato del campo de entrada	Tipo de campo de entrada (P1_WU_STE1)	Configurable

Tabla 7.2 Derivación de la propiedad de la forma de uso *Especificar el tipo de visualización del campo de entrada* el tipo de entrada a partir de la guía de captura de requisitos

La pregunta de la guía de captura de requisitos de la que se deriva la propiedad **Tipo de campo de entrada (P1_WU_STE1)** es, *¿Cuál es el formato del campo de entrada?* Esta propiedad tiene como objetivo el especificar cómo se mostrará el campo de entrada al usuario. El valor de esta propiedad lo debe determinar el analista y, por lo tanto, es una propiedad configurable. Los posibles tipos de campos entre los que puede seleccionar el analista son: checkbox, radiobutton, listbox, combobox.

Como ejemplo para ilustrar la aplicación de esta propiedad se ha seleccionado el servicio alta de cliente del sistema de alquiler de coches. La Figura 5.1 muestra una captura de pantalla de una posible interfaz para dicho servicio. Si nos centramos en el campo *Posee carné de conducir*, se puede apreciar que el valor de la propiedad *Tipo de campo de entrada* toma el valor *Checkbox*. Las otras alternativas para esta propiedad se pueden apreciar en la Figura 7.1: Radiobutton (a); Listbox (b); Campo editable (c). El uso de un modo de visualización u otro dependerá de los requisitos del usuario.

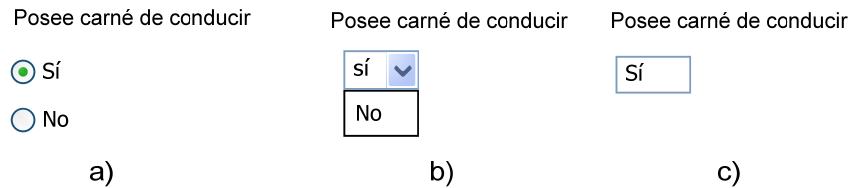


Figura 7.1 Ejemplo de tipos de valores para la propiedad *Tipo de campo de entrada*

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿En qué campos se deben introducir los datos con un formato específico?	Determinar los campos de entrada sobre los que hay que definir una máscara	Campo al que se aplicará la máscara (P1_WU_STE2)	Configurable
¿Cuál es el formato de los campos de entrada?	Definir el formato de los datos que se van a introducir mediante la máscara	Expresión regular (P2_WU_STE2)	Configurable

Tabla 7.3 Derivación de propiedades de la forma de uso *Definición de máscaras* a partir de la guía de captura de requisitos

En cuanto a las **propiedades de Definición de máscaras (WU_STE2)** definen la máscara a aplicar sobre los campos de entrada de la interfaz. La Tabla 7.3 presenta la pregunta de la guía de captura de requisitos de la que se ha derivado la propiedad y sus objetivos.

Más detalladamente, las propiedades de WU_STE2 son:

- **Campo al que se aplicará la máscara (P1_WU_STE2):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es *¿En qué campos se deben introducir los datos con un formato específico?* El objetivo de esta propiedad es el de especificar de entre todos los campos de una interfaz, cuales van a llevar asociada una máscara. Es el analista el que debe proporcionar valor a esta propiedad, por lo tanto, es una propiedad configurable.
- **Expresión regular (P2_WU_STE2):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es *¿Cuál es el formato del campo de entrada?* Esta propiedad tiene como objetivo definir el formato de los datos que se deben introducir en el sistema. Sólo los datos que cumplan esta expresión regular se podrán introducir en el sistema. La expresión regular la define el analista, por lo que esta propiedad es configurable.

Como ejemplo para ilustrar el uso de estas propiedades, se ha elegido el servicio alta de cliente del sistema de alquiler de coches. Por un lado, la propiedad *Campo al que se aplicará la máscara (P1_WU_STE2)* toma el valor *Fecha de nacimiento*. Por otro lado, la propiedad *Expresión regular (P2_WU_STE2)* especificará el formato de fecha requerido. En la Figura 7.2 se muestran algunos ejemplos de posibles máscaras: dd/mm/aaaa (a); dd-mm-aaaa (b); dd-mm-aa (c).

Fecha de nacimiento	Fecha de nacimiento	Fecha de nacimiento
<input type="text" value="07/11/1981"/>	<input type="text" value="07-11-1981"/>	<input type="text" value="07-11-81"/>
a) dd/mm/aaaa	b) dd-mm-aaaa	c) dd-mm-aa

Figura 7.2 Ejemplo de tipos de valores para la propiedad *Expresión regular*

Por último, las **propiedades de Valores por defecto (WU_STE3)** se utilizan para especificar valores por defecto que informen al usuario sobre qué formato de datos se espera recibir en un campo. La Tabla 7.4 muestra una visión global de las preguntas de la guía de captura de requisitos de las que se han derivado las propiedades y sus objetivos.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿En qué campos se deben introducir los datos con un formato específico?	Seleccionar campos que mostrarán valores por defecto	Selección del campo (P1_WU_STE3)	Configurable
¿Cuál es el formato de los campos de entrada?	Definir el valor por defecto	Definición del valor por defecto (P2_WU_STE3)	Configurable

Tabla 7.4 Derivación de propiedades de la forma de uso *Valores por defecto* a partir de la guía de captura de requisitos

A continuación se detallan cada una de las propiedades de la Tabla 7.4:

- **Selección del campo (P1_WU_STE3):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos *¿En qué campos se deben introducir los datos con un formato específico?* El objetivo de esta propiedad es el de seleccionar de entre todos los campos que forman la interfaz, cuáles de ellos tendrán definido un valor por defecto.
- **Definición del valor por defecto (P2_WU_STE3):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Cuál es el formato de los campos de entrada?* El objetivo de esta propiedad es el de especificar el valor por defecto para cada

uno de los campos de entrada seleccionados en la propiedad anterior.

Como ejemplo de uso de las propiedades de valores por defecto, podemos tomar el campo *Fecha de nacimiento* del servicio alta de cliente. Cada vez que el usuario abra la interfaz se mostrará el valor por defecto 10/12/1990, que el usuario podrá cambiar a la fecha correcta. La propiedad *Selección del campo (P1_WU_STE3)* tomaría el valor *Fecha de nacimiento* y la propiedad *Definición del valor por defecto (P2_WU_STE3)* tomaría el valor 10/12/1990.

A continuación se explica cómo afecta la incorporación de todas estas propiedades a OO-Method.

7.1.3 Modificación de los modelos conceptuales de OO-Method

De las tres formas de uso en las que se divide el mecanismo de usabilidad *Structured Text Entry*, la única que incorpora cambios a OO-Method es *Especificar el tipo de visualización del campo de entrada (WU_STE1)*. Actualmente en OO-Method, es el compilador de modelos el encargado de asignar el tipo del campo de entrada dependiendo del tipo de argumento que se introduce en cada campo. Por ejemplo, si el argumento es de tipo booleano, el compilador de modelos lo implementa mediante un checkbox, y si es de tipo cadena o entero lo representa mediante un campo editable. El analista no tiene actualmente la posibilidad de cambiar libremente el tipo del campo de entrada. La única variedad de los campos de entrada está en los argumentos objeto valuados. Para estos argumentos se permite utilizar un Combobox que lista los posibles valores para el campo o un botón que abra una UIP en la que seleccionar el argumento.

Aplicando la forma de uso WU_STE1 a OO-Method, el analista tiene la posibilidad de modificar el tipo de campo de entrada dependiendo de los requisitos del usuario. Por ejemplo, si el argumento es booleano, el analista puede elegir entre la implementación con checkbox o con radiobuttons, dependiendo de las exigencias del usuario. Por lo tanto, el único cambio que incorpora **Especificar el tipo de visualización del campo de entrada (WU_STE1)** es que el analista debe poder decidir el tipo de componente de

ventana con el que se representará cada uno de los argumentos de la interfaz.

Las otras dos formas de uso, *Definición de máscaras (WU_STE2)* y *Valores por defecto (WU_STE3)*, ya están actualmente soportadas en OO-Method. Por un lado, WU_STE2 está soportada por medio del Patrón Elemental de Introducción. Por otro lado, WU_STE3 ya está soportada a través del Modelo de Objetos. En dicho modelo se puede especificar un valor por defecto para cada uno de los argumentos que forman parte de un servicio.

La Tabla 7.5 muestra de forma resumida el único cambio que el mecanismo de usabilidad *Structured Text Entry* introduce en OO-Method. En las siguientes secciones se detallan los cambios a nivel de modelado conceptual y de compilador de modelos que son necesarios para dar soporte a este mecanismo de usabilidad.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Especificar el tipo de visualización del campo de entrada (WU_STE1)	Tipo de campo de entrada (P1_WU_STE1)	Configurable	El analista debe decidir el tipo del campo de entrada que representa al argumento

Tabla 7.5 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Structured Text Entry*

7.1.3.1 WU_STE1: Especificar el tipo de visualización del campo de entrada

La forma de uso *Especificar el tipo de visualización del campo de entrada (WU_STE1)* es la única que incorpora cambios en OO-Method. Esta forma de uso tiene una única propiedad configurable, por lo tanto, es necesario incorporar cambios en el modelado conceptual para que el analista pueda adaptar esta propiedad a los requisitos del usuario.

La propiedad indica la manera en la que se visualizarán los campos de entrada en la interfaz. Es por tanto coherente que esta propiedad se modele en la vista del modelo conceptual de OO-Method dedicado a representar cómo se visualizarán las interfaces: el **Modelo de Interacción Concreto**.

En este modelo se deben crear nuevas primitivas de modelado con las cuales especificar los aspectos visuales de los campos de entrada de información de las interfaces. Los posibles valores que pueden tomar estas primitivas de modelado dependen del tipo de argumento que se espera recibir en cada campo. A continuación se presenta la lista de nuevas primitivas conceptuales para cada campo dependiendo del tipo de argumento. Estas nuevas primitivas son sólo para argumentos de tipos simples, ya que existen actualmente primitivas conceptuales para la selección del tipo de argumentos objeto valuados.

- Booleanos:
 - Mostrar con Radiobutton
 - Mostrar con Checkbox
- Enumerado:
 - Mostrar con ListBox
 - Mostrar con ComboBox
 - Mostrar con Radiobutton
- Resto de tipos: Campo de texto editable (pensado para aquellos casos en los que se pueden recibir valores muy diversos).

Los valores por defecto para estas primitivas conceptuales son: representación de booleanos mediante checkbox; representación de enumerados mediante ListBox; representación con un campo editable para el resto de tipos.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Concreto	Indicar el tipo de campo con el que se representará cada argumento

Tabla 7.6 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Tipo de campo de entrada*

La Tabla 7.6 muestra en resumen el único cambio que incorpora *Structured Text Entry* al modelado conceptual de OO-Method.

Una vez detectados los cambios a nivel conceptual, el último paso para incorporar el mecanismo de usabilidad a OO-Method es detectar los cambios en el compilador de modelos para dar soporte a las nuevas primitivas conceptuales.

7.1.4 Modificación del compilador de modelos

7.1.4.1 WU_STE1: Tipo de campo de entrada

Los cambios realizados a nivel conceptual por la forma de uso *Especificar el tipo de visualización del campo de entrada (WU_STE1)* implican cambios en la estrategia de generación de código. Las propiedades de WU_STE1 sólo contienen aspectos visuales de la interfaz y, por lo tanto, no hay que añadir nuevos métodos al código generado, sino nuevos atributos.

La única clase del código generado que se ve afectada por la incorporación de estas propiedades es la clase *FormX* de la Figura 7.2. Los únicos cambios que incorpora *Structured Text Entry* son nuevos atributos para indicar el tipo (*TypeWidget*) y la posición (*PositionWidget*) de cada campo de entrada.

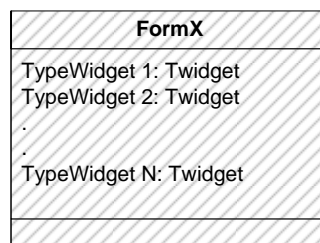


Figura 7.2 Diagrama de Clases para *Structured Text Entry*

WU_STE1 no lleva asociado ningún Diagrama de Secuencia porque su incorporación en la estrategia de generación de código no afecta a ninguna invocación de servicios.

La Tabla 7.7 muestra un resumen de los cambios que incorpora WU_STE1 al compilador de modelos de OO-Method.

Clase Genérica	Cambios que Incorpora
Form X	Añade nuevos atributos con los que representar las posibilidades de visualización de los campos de entrada de datos

Tabla 7.7 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Tipo de campo de entrada*

7.1.4.2 Resumen de los cambios que provoca Structured Text Entry

La Tabla 7.8 muestra de manera resumida las formas de uso del mecanismo de usabilidad *Structured Text Entry*, las propiedades en las que se divide, los modelos conceptuales afectados, las nuevas primitivas conceptuales que se deben añadir y los cambios en el compilador de modelos para generar todo el código que implemente el mecanismo de usabilidad.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Especificar el tipo de visualización del campo de entrada	Tipo de campo de entrada	Modelo de Interacción Concreto	El analista debe poder especificar el tipo del campo de entrada	<i>Form X</i> añade nuevos atributos con los que representar el tipo de los campos de entrada
Definición de máscaras	Campo al que se aplicará la máscara	Ninguno	Ya está soportado	Ninguno

	Expresión regular	Ninguno	Ya está soportado	Ninguno
Valores por defecto	Selección del campo	Ninguno	Ya está soportado	Ninguno
	Definición del valor por defecto	Ninguno	Ya está soportado	Ninguno

Tabla 7.8 Resumen de los cambios para incorporar *Structured Text Entry*

Capítulo 8

Incorporación de User Profile

User Profile es un FUF que tiene como objetivo permitir al usuario que personalice la aplicación de acuerdo a sus gustos. A continuación se presenta una sección por cada uno de los dos mecanismos de usabilidad en los que se divide: *Preferences* y *Personal Object Space*. En cada sección se aplica el método MIMAT a cada uno de estos mecanismos de usabilidad.

8.1 Preferences

Este patrón se utiliza para especificar qué aspectos visuales se pueden personalizar por usuario. Este tipo de personalización está especialmente pensada para aplicaciones Web, donde los perfiles de usuario son muy variados y la personalización es un aspecto muy importante para satisfacer los requisitos del usuario. Aun así, es un concepto que también se puede utilizar sobre aplicaciones de Escritorio. Este mecanismo permite a los usuarios adaptar los sistemas según sus preferencias y guardar dicha información. De esta forma, en futuras interacciones el sistema ya conoce cuales son las preferencias del usuario. El objetivo de este mecanismo de usabilidad coincide con el patrón de usabilidad de Welie llamado *Preferencias* [Welie, 2000]

8.1.1 Formas de uso

A partir de la guía de captura de requisitos del mecanismo de usabilidad *Preferences* (Anexo I) se obtienen dos formas de uso:

1. WU_P1: Preferencias en el aspecto visual.
2. WU_P2: Preferencias en el idioma.

La Tabla 8.1 contiene las preguntas de la guía de captura de requisitos de la que se han derivado las formas de uso y qué objetivo se pretende conseguir con cada una de ellas.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿El sistema proporcionará al usuario la oportunidad de adaptar el sistema a sus preferencias?	El usuario debe poder adaptar aspectos visuales de las interfaces según sus preferencias	Preferencias en el aspecto visual (WU_P1)
¿El sistema proporcionará al usuario la oportunidad de adaptar el sistema a sus preferencias?	El usuario debe poder seleccionar el idioma del sistema	Preferencias en el idioma (WU_P2)

Tabla 8.1 Derivación de las formas de uso de *Preferencias* a partir de la guía de captura de requisitos

La primera de las formas de uso, **Preferencias en el aspecto visual (WU_P1)** se deriva de la pregunta de la guía de captura de requisitos *¿El sistema proporcionará al usuario la oportunidad de adaptar el sistema a sus preferencias?* El objetivo de esta forma de uso es dar la posibilidad de personalizar aspectos relacionados con la visualización de elementos de la interfaz y el uso del sonido. Además, la característica de guardar estas preferencias es útil para que en futuras interacciones con el sistema, el usuario no tenga que volver a dedicar tiempo a personalizarla a su gusto.

Por ejemplo, en el sistema de alquiler de coches se podría usar esta forma de uso para que el usuario pudiera seleccionar sus preferencias en la

interfaz que muestra el listado de coches disponibles para alquilar. Esta funcionalidad será una de las más utilizadas por el usuario y por tanto es crítico que éste se encuentre a gusto en su uso. Una de las maneras para conseguir este propósito es darle al usuario la posibilidad de personalizar el listado.

La otra forma de uso, **Preferencias en el idioma (WU_P2)**, se deriva de la misma pregunta de la guía de captura de requisitos que WU_P1, *¿El sistema proporcionará al usuario la oportunidad de adaptar el sistema a sus preferencias?* El objetivo de esta forma de uso se diferencia de WU_P1 en que pretende personalizar el idioma del sistema y no simple aspectos visuales. El texto que se visualiza en el sistema debe estar disponible en varios idiomas, y es el usuario el que hace la elección entre ellos. Para el ejemplo del sistema de alquiler de coches, esta forma de uso es indispensable si la empresa tiene clientes a nivel internacional. Cada sucursal debería poder adaptar el idioma del sistema al idioma del país.

En la siguiente sección se explican las propiedades que hay dentro de cada una de las dos formas de uso definidas.

8.1.2 Propiedades

Esta sección explica las propiedades de las dos formas de uso en las que se divide el mecanismo de usabilidad *Preferences*. Por un lado, las **propiedades de Preferencias en el aspecto visual (WU_P1)** se utilizan para especificar las preferencias sobre aspectos visuales de las interfaces.

La Tabla 8.2 muestra una visión global de la preguntas de la guía de captura de requisitos de las que se han derivado las propiedades y el objetivo que se pretende conseguir con cada una de ellas. La pregunta es la misma para todas las propiedades, aunque cada propiedad tiene un objetivo distinto.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué preferencias serán diferentes para cada usuario?	Habilitar que el usuario pueda seleccionar el formato del texto	Selección del formato del texto (P1_WU_P1)	Configurable
¿Qué preferencias serán diferentes para cada usuario?	Habilitar que el usuario pueda seleccionar los iconos	Selección de iconos (P2_WU_P1)	Configurable
¿Qué preferencias serán diferentes para cada usuario?	Habilitar que el usuario pueda decidir qué partes del sistema visualizar	Selección de los elementos a visualizar (P3_WU_P1)	Configurable
¿Qué preferencias serán diferentes para cada usuario?	Habilitar que el usuario pueda seleccionar los colores del sistema	Selección de Colores (P4_WU_P1)	Configurable
¿Qué preferencias serán diferentes para cada usuario?	Habilitar que el usuario pueda decidir si desea oír el sonido del sistema	Uso de sonido (P5_WU_P1)	Configurable

Tabla 8.2 Derivación de propiedades de la forma de uso *Preferencias en el aspecto visual* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades de WU_P1 son las siguientes:

- **Selección del formato del texto (P1_WU_P1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué preferencias serán diferentes para cada usuario?* El objetivo de esta propiedad es el de habilitar la posibilidad de que el usuario pueda adaptar el formato del texto (tamaño, estilo). El formato indicado por el analista es el formato por defecto, pero el usuario final debería poder modificarlo según sus preferencias. El analista es el que habilita o inhabilita esta preferencia, por lo tanto esta propiedad es configurable.
- **Selección de iconos (P2_WU_P1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué preferencias serán diferentes para cada usuario?* El objetivo de esta propiedad es el de habilitar la posibilidad de que el usuario pueda seleccionar los iconos del sistema según sus preferencias. El analista puede bien definir una lista con unos cuantos iconos para que el usuario elija entre ellos o bien dar libertad al usuario para que utilice iconos propios. El analista es el que habilita o inhabilita esta preferencia, por lo tanto esta propiedad es configurable.
- **Selección de los elementos a visualizar (P3_WU_P1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué preferencias serán diferentes para cada usuario?* El objetivo de esta propiedad es que el usuario pueda elegir qué elementos desea ver de una interfaz y cuáles ocultar. Además, esta propiedad también debe permitir modificar las etiquetas de los elementos de la interfaz. El analista es el que habilita o inhabilita esta preferencia, por lo tanto esta propiedad es configurable.
- **Selección de colores (P4_WU_P1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué preferencias serán diferentes para cada usuario?* El objetivo de esta propiedad es dotar al usuario de la capacidad de personalizar los colores usados en el sistema. El analista es el que habilita o inhabilita esta preferencia, por lo tanto, esta propiedad es configurable

- **Uso de sonido (P5_WU_P1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué preferencias serán diferentes para cada usuario?* El objetivo de esta propiedad es permitir al usuario que pueda decidir cuándo desea oír el sonido del sistema y cuando no. Esta propiedad es también configurable, ya que el analista habilita o inhabilita esta funcionalidad.

Como ejemplo para ilustrar la utilidad de estas propiedades, se ha elegido la funcionalidad de mostrar el listado de coches del sistema de alquiler de coches. Una posible interfaz para esta funcionalidad podría ser la mostrada en la Figura 8.1. Esta interfaz es la que se mostraría por defecto antes de que el usuario hubiera seleccionado sus preferencias. En caso de que el analista hubiera habilitado todas las propiedades que conforman la forma de uso *Preferencias en el aspecto visual*, el usuario podría adaptar el aspecto de esta interfaz a sus preferencias.

La Figura 8.2 representa la misma interfaz que lista los coches pero en la cual el usuario ha aplicado sus preferencias. Las preferencias para este ejemplo hubieran sido: cambiar el formato de la etiqueta *Marca*; añadir un icono al botón que ejecuta el filtrado; ocultar la columna *Radio CD* de la tabla; modificar los colores de la tabla.

Marca	Modelo	Motor	Color	Caballos	Aire Acond.	Radio CD	Num puertas
SEAT	Ibiza	Gasolina	Rojo	80	No	Sí	3
SEAT	Ibiza	Diesel	Negro	120	Sí	Sí	3
SEAT	León	Gasolina	Blanco	105	Sí	Sí	5
FORD	Focus	Diesel	Plata	120	Sí	No	5
FORD	Fiesta	Diesel	Rojo	80	No	No	5
CITROEN	C3	Gasolina	Azul	80	No	Sí	5

Figura 8.1 Listado de coches sin preferencias

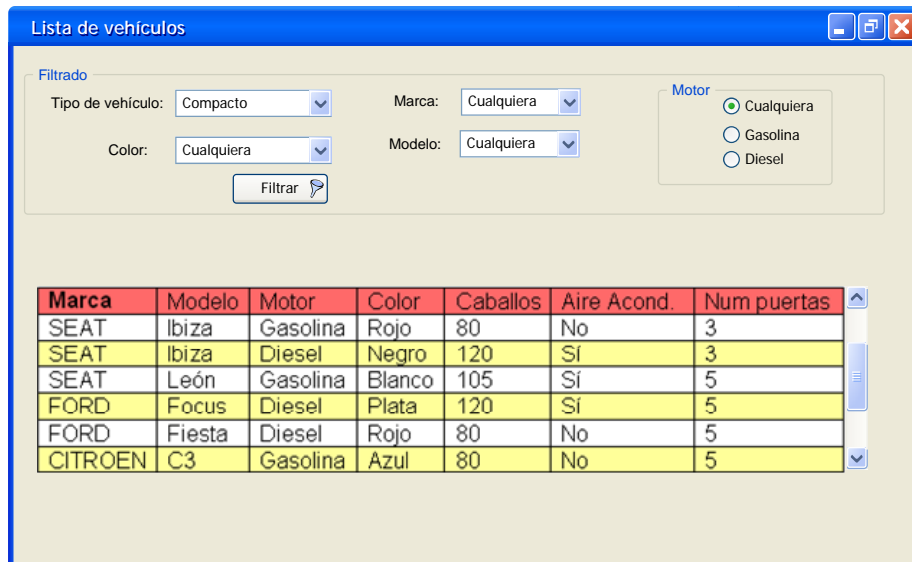


Figura 8.2 Listado de coches con preferencias

Por último, la forma de uso **Preferencias en el idioma (WU_P2)** tiene una única propiedad mediante la cual el analista habilita al usuario para que pueda adaptar el idioma del sistema al suyo propio. La Tabla 8.3 muestra la pregunta de la guía de captura de requisitos utilizada para obtener la propiedad y el objetivo que se pretende alcanzar con ella.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué preferencias serán diferentes para cada usuario?	Permitir adaptar el sistema al lenguaje del usuario	Preferencias en el idioma (P1_WU_P2)	Configurable

Tabla 8.3 Derivación de la propiedad de la forma de uso *Preferencias en el idioma* a partir de la guía de captura de requisitos

La **propiedad Preferencias en el idioma** se deriva de la pregunta de la guía de captura de requisitos, *¿Qué preferencias serán diferentes para cada*

usuario? El objetivo de esta propiedad es adaptar el idioma del sistema al idioma del usuario. Tomando como ejemplo el listado de coches del sistema de alquiler, el usuario podría modificar el idioma de las etiquetas del sistema si el analista habilitara esta funcionalidad mediante la propiedad *Preferencias en el idioma*. Es importante remarcar que el cambio de idioma sólo afectaría a etiquetas del sistema y no a la información almacenada en la base de datos. La Figura 8.3 muestra un ejemplo de cambio de idioma.

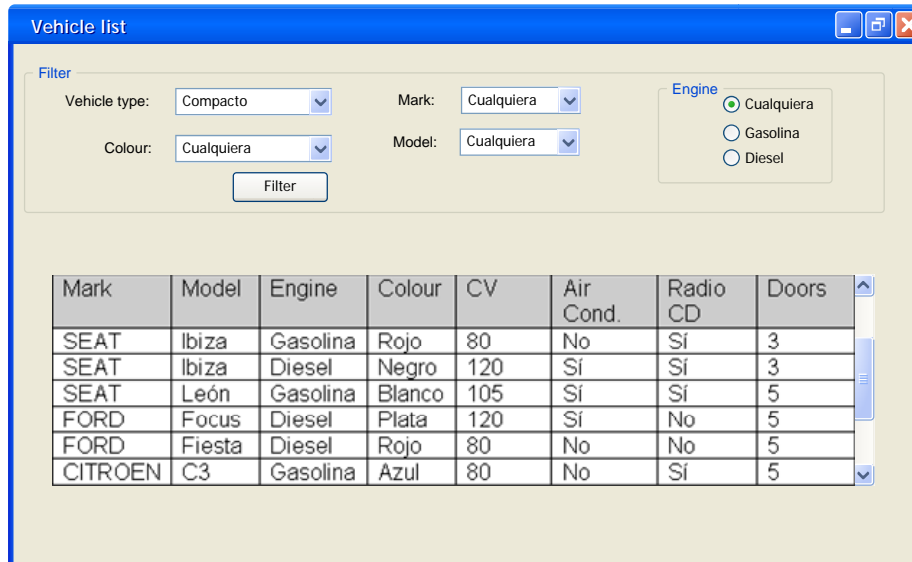


Figura 8.3 Listado de coches cambiando el idioma

A continuación se van a explicar los cambios que son necesarios realizar en OO-Method para incorporar las propiedades de las dos formas de uso del mecanismo de usabilidad *Preferences*.

8.1.3 Modificación de los modelos conceptuales de OO-Method

Tanto la forma de uso *Preferencias en el aspecto visual (WU_P1)* como *Preferencias en el idioma (WU_P2)* tienen una aplicación práctica sobre los sistemas generados con OO-Method. La única propiedad que no tiene sentido aplicar a los sistemas generados con OO-Method es *Uso del sonido (P5_WU_P1)*, ya que las aplicaciones generadas no disponen de ningún

sonido. El resto de propiedades de **Preferencias en el aspecto visual (WU_P1)** tienen aplicación en los sistemas de OO-Method e incluso existen algunas propiedades que están parcialmente soportadas por OO-Method: *Selección del formato del texto (P1_WU_P1)* y *Selección de elementos a visualizar (P3_WU_P1)*. Los últimos cambios que se han incorporado a OO-Method han ido orientados a permitir la personalización de algunos elementos de las interfaces. En concreto, ya se soportan algunas características de la propiedad *Selección de elementos a visualizar (P3_WU_P1)*:

- El usuario puede decidir los atributos visibles en cada columna de una interfaz generada a partir de una UIP o UII.
- El usuario puede decidir las etiquetas (alias) de las columnas de una interfaz generada a partir de una UIP o UII.

Tal y como se puede apreciar, todas las posibilidades de personalización afectan a las UIPs y UIIs. Deben aplicar las propiedades de WU_P1 también a las ventanas generadas a partir de una UIS. Además, hay que enriquecer las posibilidades de personalización de las aplicaciones generadas por OO-Method con las propiedades aun no incluidas y aspectos aun no soportados de las propiedades *Selección del formato del texto* y *Selección de elementos a visualizar*. Otro aspecto que se puede mejorar, es que actualmente es el compilador de modelos el que incluye la posibilidad de personalización sin la decisión del analista, por tanto son propiedades no configurables. En cambio, puede darse el caso de aplicaciones que debido a varios motivos (falta de experiencia de los usuarios, dominio de la aplicación, etc.) no requieran ninguna personalización. Por ello, es recomendable que la personalización de las interfaces sea un aspecto configurable por parte del analista. Por tanto, los cambios que se deben llevar a cabo en OO-Method para soportar las propiedades de WU_P1 son:

- El analista debe poder decidir si desea que el usuario modifique el formato del texto de las interfaces.
- El usuario debe poder modificar el formato de los elementos de las interfaces.

- El analista debe poder decidir si desea que el usuario modifique los iconos del sistema.
- El usuario debe poder modificar los iconos del sistema.
- El analista debe poder decidir si desea que el usuario modifique los elementos a visualizar.
- El usuario debe poder ocultar y volver a visualizar elementos del sistema.
- El analista debe poder decidir si desea que el usuario modifique los colores del sistema.
- El analista debe poder modificar los colores del sistema.

Por otro lado, la forma de uso **Preferencias en el idioma (WU_P2)** ya está actualmente soportada por OO-Method, por lo tanto no incorpora ningún cambio. El conjunto de herramientas de OlivaNOVA incluye una para gestionar los distintos tipos de idiomas de una aplicación. Esta herramienta se llama *OlivaNOVA Multilanguage Manager*.

La Tabla 8.4 muestra de forma resumida los cambios que incorporan a OO-Method cada una de las propiedades del mecanismo de usabilidad *Preferences*. En la siguiente sección se explican los cambios que es necesario llevar a cabo en el modelado conceptual para soportar las propiedades configurables de WU_P1.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Preferencias en el aspecto visual (WU_P1)	Selección del formato del texto (P1_WU_P1)	Configurable	<ul style="list-style-type: none"> – El analista debe poder habilitar las preferencias en el formato del texto – El usuario debe poder elegir el formato del texto

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
	Selección de iconos (P2_WU_P1)	Configurable	<ul style="list-style-type: none"> – El analista debe poder habilitar las preferencias en los iconos – El usuario debe poder elegir los iconos
	Selección de los elementos a visualizar (P3_WU_P1)	Configurable	<ul style="list-style-type: none"> – El analista debe poder habilitar las preferencias en los elementos visibles – El usuario debe poder seleccionar qué elementos visualizar
	Selección de colores (P4_WU_P1)	Configurable	<ul style="list-style-type: none"> – El analista debe poder habilitar las preferencias en los colores – El usuario debe poder elegir los colores del sistema
Preferencias en el idioma (WU_P2)	Preferencias en el idioma (P1_WU_P2)	Configurable	<ul style="list-style-type: none"> – El analista debe poder habilitar el cambio de idioma – El usuario debe poder elegir el idioma del sistema

Tabla 8.4 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Preferences*

8.1.3.1 WU_P1: Preferencias en el aspecto visual

Todas las propiedades derivadas de la forma de uso *Preferencias en el aspecto visual (WU_P1)* son configurables, y por lo tanto, deben disponer de primitivas conceptuales para representarlas. Aunque las aplicaciones generadas con OO-Method ya disponen de varios elementos personalizables, el analista no tiene la posibilidad de decidir qué elementos serán personalizables y cuales no. Actualmente, los elementos personalizables son los mismos en todos los sistemas desarrollados. Por lo tanto, la incorporación de nuevas primitivas conceptuales se realiza para que sea el analista el responsable de dotar de esta funcionalidad a los componentes de las interfaces. Los cambios que se deben incorporar al modelado conceptual afectan al cómo se va a visualizar la información en las interfaces, y por tanto, se deben modelar las nuevas primitivas conceptuales en el **Modelo de Interacción Concreto**. En este modelo, el analista puede seleccionar para cada UI qué aspectos de la interfaz serán o no personalizables. Para ello, se define una primitiva por cada una de las personalizaciones que el analista puede habilitar:

- El formato del texto de las interfaces necesita dos Primitivas:
 - Primitiva para representar si la personalización del tamaño está habilitada
 - Primitiva para representar si la personalización del estilo está habilitada
- Si los iconos de los botones de las interfaces son o no modificables. En caso de que sean modificables, el analista puede determinar un conjunto de iconos posibles. Son necesarias dos primitivas mediante las cuales el analista puede habilitar o inhabilitar la personalización de los iconos de los siguientes elementos:
 - Botones de acción de las UIPs y UIIs
 - Botones para la ejecución de un servicio en una UIS (OK y Cancel)
- El analista puede determinar si permite que el usuario pueda decidir sobre los siguientes aspectos relacionados con la visualización de elementos:

- La visibilidad de los campos de entrada de una UIS. Solo se podrán ocultar aquellos campos que no sean obligatorios
- La modificación de las etiquetas (alias) de los campos de una UIS
- La visibilidad de las acciones de una UIP o UII
- La modificación de las etiquetas de las acciones en las UIPs y UIIs
- La visibilidad de las navegaciones de una UIP
- La modificación de las etiquetas de las navegaciones de las UIPs
- La visibilidad de los filtros de una UIP
- La modificación de las etiquetas de los filtros de una UIP
- La visibilidad de las variables de filtro de una UIP
- La modificación de las etiquetas de las variables de filtro de una UIP
- La visibilidad de los criterios de ordenación de una UIP
- La modificación de las etiquetas de los criterios de ordenación de una UIP
- El analista debe poder habilitar o inhabilitar el cambio de color de los elementos de la interfaz. Estos cambios afectan a las interfaces de toda la aplicación para mantener la homogeneidad y no contradecir el criterio ergonómico de *consistencia* [Bastien, 1993]. Serán necesarias tres primitivas para representar si se habilita la personalización de los colores de los siguientes aspectos:
 - El fondo de las interfaces
 - El fondo de las tablas
 - La línea de pijama de las tablas (línea en distinto color cada dos filas para facilitar su lectura)

El valor por defecto de estas nuevas primitivas es que habilitan al usuario la personalización de todos los elementos de la interfaz.

La Tabla 8.5 presenta a modo de resumen los cambios que incorpora *Preferences* al modelado conceptual de OO-Method.

Modelo de OO-Method	Nueva Primitiva Conceptual
<p>Modelo de Interacción Concreto</p>	<ul style="list-style-type: none"> – Habilitar personalización del tamaño del texto de la interfaz – Habilitar personalización del estilo del texto de la interfaz – Habilitar cambio de iconos en acciones de UI de Población y UI de Instancia – Habilitar cambios de iconos en botones de UI de Servicio – Habilitar ocultar campos de entrada – Habilitar modificar alias de campos de entrada – Habilitar ocultar acciones de una UI de Población o UI de Instancia – Habilitar modificar alias de las acciones de una UI de Población o UI de Instancia – Habilitar ocultar navegaciones de una UI de Población o UI de Instancia – Habilitar modificar alias de las navegaciones de una UI de Población – Habilitar ocultar filtros de una UI de Población – Habilitar modificar alias de los filtros de una UI de Población – Habilitar ocultar variables de filtro de una UI de Población – Habilitar modificar alias de las variables de filtro de una UI de Población – Habilitar ocultar criterios de ordenación de una UI de Población – Habilitar modificar los alias de los criterios de ordenación de una UI de Población

Modelo de OO-Method	Nueva Primitiva Conceptual
	<ul style="list-style-type: none"> – Habilitar el cambio de color del fondo de las interfaces – Habilitar el cambio de color del fondo de las tablas – Habilitar línea de pijama en las tablas

Tabla 8.5 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Preferencias en el aspecto visual*

En la siguiente sección se explican los cambios que se deben llevar a cabo en el compilador de modelos para la incorporación de las propiedades de *Preferencias en el aspecto visual*.

8.1.4 Modificación del compilador de modelos

8.1.4.1 WU_P1: Preferencias en el aspecto visual

Las nuevas primitivas conceptuales derivadas de la forma de uso *Preferencias en el aspecto visual (WU_P1)* implican cambios en el compilador de modelos, que debe ser capaz de generar el código que las implemente.

El Diagrama de Clases de la Figura 8.4 representa los nuevos métodos y atributos que se deben añadir a las clases generadas por el compilador de modelos para dar soporte a las propiedades. Sólo se ven afectadas las tres clases que representan a las tres UIs del Modelo de Interacción Abstracto de OO-Method: *FormX*, para la personalización de los elementos que componen una UIS; *FrmGnPopulation*, para la personalización de los elementos que componen una UIP; *FrmGnInstance*, para la personalización de los elementos que componen una UII.

Cada una de estas clases tiene un método por cada una de las propiedades que componen la forma de uso WU_P1. Mediante estos métodos, el usuario puede personalizar los elementos que componen cada una de las

propiedades (siempre que el analista haya habilitado su personalización mediante las primitivas conceptuales). Aunque los métodos se llaman igual en las tres clases, su implementación es distinta. Por ejemplo, el método *Personalize_icons* personaliza los iconos de los botones *OK* y *Cancel* en una *UIS* mediante la clase *FormX*, mientras que en la clase *FrmGnPopulation* personaliza los iconos de los botones de acción de una *UIP*.

Los nuevos métodos modificarán el valor de los atributos que representan las características personalizables de la interfaz. Actualmente ya se generan atributos para representar las características de la interfaz, por lo que no hay que añadir nuevos atributos a las clases generadas. Existe por tanto un atributo por cada elemento personalizable de la interfaz que almacenará la personalización elegida por el usuario. En el diagrama de la Figura 8.4 aparecen atributos en cada clase para representar: el formato del texto de las interfaces (*Format_textX*); el icono de los botones (*IconX*); la visibilidad de los elementos de la interfaz (*Visibility_elemX*); el color de fondo de la interfaz y las tablas (*TColourX*). Todos estos atributos tienen la letra *X* en su nombre porque existirá uno de estos atributos por cada elemento de la interfaz que tenga ese atributo personalizable.

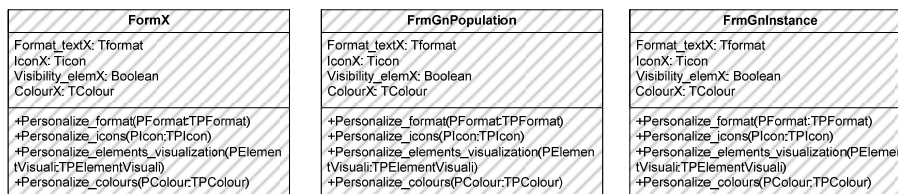


Figura 8.4 Diagrama de Clases para *Preferences*

Utilizando las clases definidas en la Figura 8.4 se puede construir un Diagrama de Secuencia para expresar la llamada de eventos entre clases (Figura 8.5). En el caso de la forma de uso *WU_P1*, no hay relación de asociación entre las tres clases que se ven modificadas por el patrón *Preferences*, por lo tanto, no hay relación en las invocaciones de métodos de las tres clases. El Diagrama de Secuencia sólo muestra que el usuario puede indistintamente invocar la personalización en las tres clases, sin que haya un orden estricto a seguir.

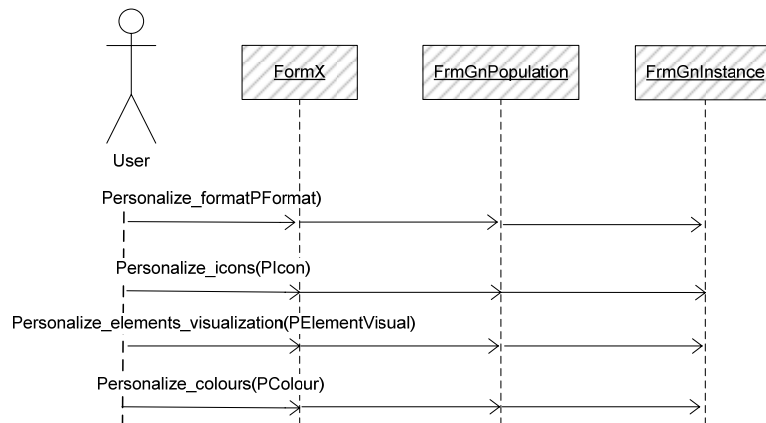


Figura 8.5 Diagrama de Secuencia para incorporar la forma de uso *Preferencias en el aspecto visual*

La Tabla 8.6 muestra un resumen de los cambios que incorporaría WU_P1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	<ul style="list-style-type: none"> – Método para personalizar el formato de la interfaz – Método para personalizar los iconos de los botones (acciones) – Método para personalizar los elementos visibles de la interfaz – Método para personalizar los colores de la interfaz
FrmGnPopulation	<ul style="list-style-type: none"> – Método para personalizar el formato de la interfaz – Método para personalizar los iconos de los botones (OK, Cancel) – Método para personalizar los elementos visibles de la interfaz – Método para personalizar los colores de la interfaz

Clase Genérica	Cambios que Incorpora
FrmGnInstance	<ul style="list-style-type: none"> – Método para personalizar el formato de la interfaz – Método para personalizar los iconos de la interfaz – Método para personalizar los elementos visibles de la interfaz – Método para personalizar los colores de la interfaz

Tabla 8.6 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Preferencias en el aspecto visual*

8.1.4.2 Resumen de los cambios que provoca Preferences

La Tabla 8.7 muestra de manera resumida las formas de uso del mecanismo de usabilidad *Preferences*, las propiedades en las que se divide, los modelos conceptuales afectados, las nuevas primitivas conceptuales que se deben añadir y los cambios en el compilador de modelos para generar el código que soporte la funcionalidad del mecanismo de usabilidad.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Preferencias en el aspecto visual	Selección del formato del texto	Modelo de Interacción Concreto	Habilitar al usuario para que pueda personalizar el formato del texto	<ul style="list-style-type: none"> – <i>Form X</i> tiene un método para personalizar el texto – <i>FrmGnPopulation</i> tiene un método para personalizar el formato

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
				– <i>FrmGnInstan</i> ce tiene un método para personalizar el formato
	Selección de iconos	Modelo de Interacción Concreto	Habilitar al usuario para que pueda seleccionar los iconos	– <i>Form X</i> tiene un método para personalizar los iconos – <i>FrmGnPopul</i> ation tiene un método para personalizar los iconos – <i>FrmGnInstan</i> ce tiene un método para personalizar los iconos
	Selección de los elementos a visualizar	Modelo de Interacción Concreto	Habilitar al usuario para que pueda ocultar y modificar el alias de campos de entrada, acciones, navegaciones, filtros, variables de filtro, criterios de	– <i>Form X</i> tiene un método para personalizar los elementos de la interfaz – <i>FrmGnPopul</i> ation tiene un método para personalizar los elementos de la interfaz

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
			ordenación	– <i>FrmGnInstance</i> tiene un método para personalizar los elementos de la interfaz
	Selección de colores	Modelo de Interacción Concreto	Habilitar al usuario para que pueda cambiar los colores de las interfaces y de las tablas	– <i>Form X</i> tiene un método para personalizar los colores – <i>FrmGnPopulation</i> tiene un método para personalizar los colores – <i>FrmGnInstance</i> tiene un método para personalizar los colores
	Uso de sonido	Ninguno	Ninguno	Ninguno
Preferencias en el idioma	Preferencias en el idioma	Ninguno	Ya soportada	Ninguno

Tabla 8.7 Resumen de los cambios para incorporar *Preferences*

8.2 Personal Object Space

Este mecanismo de usabilidad se utiliza para incorporar a una aplicación la posibilidad de que el usuario pueda modificar la posición de los elementos de la interfaz según sus preferencias. Esta funcionalidad ayuda al usuario a recordar más fácilmente dónde se encuentran los elementos de la interfaz. Al igual que el mecanismo de usabilidad *Preferences*, *Personal Object Space* está especialmente pensado para aplicaciones Web, donde los usuarios provienen de distintos ámbitos. Aun así, también se debe poder utilizar en aplicaciones de Escritorio. El objetivo de este mecanismo de usabilidad coincide con el del patrón de usabilidad *Organización de la información* de Welie [Welie, 2000].

8.2.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *Personal Object Space* (Anexo I) se deriva una única forma de uso, **Distribución de elementos (WU_POS1)**. La Tabla 8.8 muestra la pregunta a partir de la que se ha derivado la forma de uso y el objetivo que se pretende conseguir con ella.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Qué elementos de la interfaz debería poder organizar el usuario?	Habilitar al usuario para que distribuya los elementos de la interfaz según sus preferencias	Distribución de elementos (WU_POS1)

Tabla 8.8 Derivación de la forma de uso de *Personal Object Space* a partir de la guía de captura de requisitos

La pregunta de la guía de captura de requisitos de la que se deriva WU_POS1 es, *¿Qué elementos de la interfaz debería poder organizar el usuario?* El objetivo de esta forma de uso es el poder dotar al usuario de la capacidad de cambiar la distribución de los elementos de las interfaces. Una vez cambiada esta distribución, se debe almacenar de alguna forma para mantenerla en futuras interacciones del usuario con el sistema.

Por ejemplo, en la ventana que muestra el listado de coches dentro del sistema de alquiler, el usuario podría ordenar los campos con información sobre el coche según sus preferencias.

8.2.2 Propiedades

Las **propiedades de Distribución de elementos (WU_POS1)** se utilizarán para especificar qué elementos se podrán distribuir por la interfaz y de qué forma. La Tabla 8.9 muestra una visión global de la pregunta de la guía de captura de requisitos de la que se han derivado todas las propiedades y cuál es el objetivo de cada una de esas propiedades. Aunque la pregunta es la misma para todas las propiedades, el objetivo de cada propiedad es distinto.

Pregunta de la guía o patrón de usabilidad	Objetivo de la propiedad	Propiedad	Tipo
¿Qué elementos de la interfaz debería poder organizar el usuario?	Habilitar que el usuario pueda mover elementos de la interfaz	Mover elementos (P1_WU_POS1)	Configurable
¿Qué elementos de la interfaz debería poder organizar el usuario?	Habilitar que el usuario pueda agrupar elementos de la interfaz	Agrupar elementos (P2_WU_POS1)	Configurable
¿Qué elementos de la interfaz debería poder organizar el usuario?	Habilitar que el usuario pueda ordenar elementos de la interfaz	Ordenar elementos (P3_WU_POS1)	Configurable

Pregunta de la guía o patrón de usabilidad	Objetivo de la propiedad	Propiedad	Tipo
¿Qué elementos de la interfaz debería poder organizar el usuario?	Habilitar que el usuario pueda seleccionar el tipo de alineación de la información	Alinear elementos (P4_WU_POS1)	Configurable
¿Qué elementos de la interfaz debería poder organizar el usuario?	Habilitar que el usuario pueda modificar el tamaño de los elementos de la interfaz	Tamaño de los elementos (P5_WU_POS1)	Configurable

Tabla 8.9 Derivación de propiedades de la forma de uso *Distribución de elementos* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades de WU_POS1 son las siguientes:

- **Mover elementos (P1_WU_POS1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué elementos de la interfaz debería poder organizar el usuario?* El objetivo de esta propiedad es que el analista decida qué elementos de la interfaz podrá mover el usuario. Esta propiedad es configurable, ya que depende de la decisión del analista.
- **Agrupar elementos (P2_WU_POS1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué elementos de la interfaz debería poder organizar el usuario?* El objetivo de esta propiedad es que el analista decida qué elementos de la interfaz podrá agrupar el usuario. Es también una propiedad configurable.
- **Ordenar elementos (P3_WU_POS1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué elementos de la interfaz debería poder organizar el usuario?* El objetivo de esta

propiedad es que el analista dote al usuario de la capacidad de ordenar los elementos de la interfaz a partir de varios criterios: el orden alfabético, el tamaño, el tipo de datos. Es una propiedad configurable.

- **Alinear elementos (P4_WU_POS1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué elementos de la interfaz debería poder organizar el usuario?* El objetivo de esta propiedad es que el analista habilite al usuario para que éste pueda alinear los elementos de la interfaz según sus preferencias (a la izquierda, centrados, a la derecha o justificados).
- **Tamaño de los elementos (P5_WU_POS1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué elementos de la interfaz debería poder organizar el usuario?* El objetivo de esta propiedad es que el analista habilite al usuario para que pueda modificar el tamaño de los elementos de la interfaz y así facilitar su interacción con el sistema.

Como ejemplo para ilustrar la aplicación de estas propiedades, se ha utilizado la interfaz del servicio listado de coches representada en la Figura 8.1. Si se supone que el analista ha habilitado al usuario para que sea capaz de distribuir los elementos de la interfaz, podría quedar una interfaz como la mostrada en la Figura 8.6. En dicha figura, el analista habría habilitado la movilidad de elementos mediante la propiedad *Mover elementos (P1_WU_POS1)*. De esta forma el usuario hubiera movido los campos de filtrado según sus preferencias. Al habilitar el analista la agrupación de elementos mediante la propiedad *Agrupar elementos (P2_WU_POS1)*, el usuario podría agrupar los filtros entre los que más usa (*Marca* y *Modelo*) y el resto. Si el analista habilitara la ordenación de elementos mediante la propiedad *Ordenar elementos (P3_WU_POS1)*, el usuario podría ordenar las columnas que forman el listado de coches según sus preferencias. En el ejemplo, ha movido la columna *Caballos* unas posiciones más a la izquierda. Si el analista habilitara la alineación de elementos mediante la propiedad *Alinear elementos (P4_WU_POS1)*, el usuario podría indicar cuál es la alineación de la lista de coches. En el ejemplo se ha elegido una alineación *Centrada*. Por último, si el analista habilitara la modificación del *Tamaño de los elementos (P5_WU_POS1)*, el usuario podría adaptar el tamaño de los campos de la tabla a sus

necesidades. En el ejemplo, se ha dado más tamaño a la columna *Aire Acond* y *Radio CD*.

Marca	Modelo	Caballos	Motor	Color	Aire Acond.	Radio CD	Num puertas
SEAT	Ibiza	80	Gasolina	Rojo	No	Sí	3
SEAT	Ibiza	120	Diesel	Negro	Sí	Sí	3
SEAT	León	105	Gasolina	Blanco	Sí	Sí	5
FORD	Focus	120	Diesel	Plata	Sí	No	5
FORD	Fiesta	80	Diesel	Rojo	No	No	5
CITROEN	C3	80	Gasolina	Azul	No	Sí	5

Figura 8.6 Listado de coches aplicando las propiedades de *Distribución de elementos*

La siguiente sección muestra cómo las propiedades configurables afectan al modelo conceptual de OO-Method.

8.2.3 Modificación de los modelos conceptuales de OO-Method

OO-Method ya soporta en parte la capacidad de que el usuario distribuya los elementos de la interfaz según sus preferencias y almacena dicha información. Las posibilidades de distribución que ofrece OO-Method son las siguientes:

- El usuario puede cambiar la posición en la que se abre cada vez una interfaz.
- En una interfaz generada a partir de una UIP, el usuario puede ordenar los campos de una misma columna mediante el patrón elemental de *Ordenación*.

- En una interfaz generada a partir de una UIP, el usuario puede modificar el tamaño de los campos que se visualizan en la tabla de instancias y también puede modificar el número de instancias mostradas en cada UIP antes de volver a cargar más datos.

Sin embargo, estas funcionalidades no son incluidas en el sistema por decisión del analista, sino que es el compilador de modelos el encargado de incorporarlas a todos los sistemas. Por tanto, una de las mejoras que habría que incluir en OO-Method es que el analista tuviera decisión sobre qué sistemas dispondrán de estas funcionalidades y cuales no.

Toda la funcionalidad que ofrece OO-Method para que el usuario modifique la disposición de elementos en la ventana afecta a elementos de las UIPs. Habría que extender estas propiedades al resto de UIs (UIs y UIIs) e incorporar aquellas propiedades aun no soportadas. Por todo ello, es necesario enriquecer el modelado conceptual de OO-Method para incorporar las propiedades de *Distribución de elementos*.

La Tabla 8.10 muestra una relación entre las propiedades de *Distribución de elementos* y los cambios que se deben introducir en OO-Method para dar soporte a dichas propiedades. En la siguiente sección se muestran en detalle las nuevas primitivas conceptuales derivadas de las propiedades configurables.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Distribución de elementos (WU_POS1)	Mover elementos (P1_WU_POS1)	Configurable	El analista debe poder habilitar al usuario para que mueva elementos de la interfaz
	Agrupar elementos (P2_WU_POS1)	Configurable	El analista debe poder habilitar al usuario para que agrupe elementos

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
	Ordenar elementos (P3_WU_POS1)	Configurable	El analista debe poder habilitar al usuario para que pueda ordenar elementos de la interfaz
	Alinear elementos (P4_WU_POS1)	Configurable	El analista debe poder habilitar al usuario para que pueda alinear la información
	Tamaño de los elementos (P5_WU_POS1)	Configurable	El analista debe poder habilitar al usuario para que pueda modificar el tamaño de los elementos de la interfaz

Tabla 8.10 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Personal Object Space*

8.2.3.1 WU_POS1: Distribución de elementos

Todas las propiedades de la forma de uso *Distribución de elementos (WU_POS1)* son configurables, y por tanto, requieren de primitivas conceptuales que las representen. Mediante las nuevas primitivas, el analista debe poder indicar sobre qué elementos de las UIs el usuario podrá aplicar cambios en la distribución de sus elementos. El único modelo de OO-Method que se ve afectado por la incorporación de esta forma de uso es el **Modelo de Interacción Concreto**. Mediante este modelo, el analista selecciona sobre qué elementos podrá el usuario cambiar su distribución

dentro de la interfaz. Cada una de las posibles habilitaciones que puede otorgar el analista al usuario da lugar a una nueva primitiva conceptual:

- El analista debe poder determinar si permite al usuario personalizar el lugar de la pantalla donde se abre cada interfaz. La funcionalidad de cambiar y almacenar dónde se abre cada interfaz ya está soportada actualmente por OO-Method.
- El analista debe poder especificar si permite la modificación de la agrupación de los campos de entrada en una US. Esta especificación se lleva a cabo a través de una nueva primitiva. El analista puede realizar una primera agrupación mediante el patrón elemental de *Agrupación*, pero se puede también habilitar al usuario para que la modifique según sus preferencias.
- El analista debe poder habilitar o inhabilitar la ordenación de los siguientes elementos:
 - Argumentos de entrada de una UIS
 - El orden de los botones de acción de las UIPs y UIIs
 - El orden de los botones de navegación de las UIPs
 - El orden de las variables de filtro de una UIP
 - El orden de los filtros de una UIP
 - El orden de los criterios de ordenación de una UIP
 - Columnas de la tabla de una UIP (ya soportada por el compilador de modelos actualmente)

La habilitación de la ordenación de cada uno de los siguientes elementos se representa mediante primitivas conceptuales. Debe existir una primitiva por cada elemento que tenga la capacidad de ordenación.

- El analista debe poder habilitar la alineación de los siguientes elementos:
 - Argumentos de entrada de una UIS
 - Variables de filtro de una UIP
- El analista debe poder habilitar la modificación del tamaño de los siguientes elementos:

- Columnas de una UIP (ya soportada por el compilador de modelos actualmente)
- Número de instancias mostradas en cada UIP antes de volver a cargar datos (ya soportada por el compilador de modelos actualmente)
- Argumentos de entrada dentro de una UIS
- Variables de filtro dentro de las interfaces derivadas de las UIPs

El valor por defecto de todas estas primitivas es que permiten al usuario distribuir los elementos de la interfaz. La Tabla 8.11 muestra un resumen de las nuevas primitivas conceptuales que se deben incorporar a OO-Method para dar soporte a la forma de uso *Distribución de elementos*.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Concreto	<ul style="list-style-type: none"> – Habilitar que el usuario modifique la posición en la que se abran las interfaces – Habilitar que el usuario agrupe elementos de las interfaces – Habilitar que el usuario ordene los campos de entrada de una UIS – Habilitar que el usuario ordene las acciones de una UIP o UII – Habilitar que el usuario ordene las navegaciones en una UI de Población – Habilitar que el usuario ordene las variables de filtro de una UIP – Habilitar que el usuario ordene los filtros de una UIP – Habilitar que el usuario ordene los criterios de ordenación de una UIP – Habilitar que el usuario alinee los campos de entrada de una UIS

Modelo de OO-Method	Nueva Primitiva Conceptual
	<ul style="list-style-type: none"> – Habilitar que el usuario alinee las variables de filtro de una UIP – Habilitar que el usuario modifique el tamaño de las columnas de una UIP – Habilitar que el usuario modifique el número de instancias de una UIP – Habilitar que el usuario modifique el tamaño de los campos de una UIS – Habilitar que el usuario modifique el tamaño de las variables de filtro de una UIP

Tabla 8.11 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Distribución de elementos*

8.2.4 Modificación del compilador de modelos

8.2.4.1 WU_POS1: Distribución de elementos

Los cambios a nivel conceptual para permitir la distribución de elementos de la pantalla implican cambios en el compilador de modelos. La Figura 8.7 muestra el Diagrama de Clases con las clases afectadas por la incorporación de las propiedades de *Distribución de elementos*. Las únicas clases afectadas son las mismas que para el caso de la incorporación del mecanismo de usabilidad *Preferences*, es decir, las clases que implementan una UIS, una UIP y una UII respectivamente. Al igual que para el mecanismo *Preferences*, aunque los métodos se llamen igual en las tres clases, su implementación tendría distinta funcionalidad. Existe un método para modificar la posición en la que se abre cada interfaz (*Move_elements*); un método para permitir la agrupación de los elementos (*Elements_grouping*); un método para alinear elementos (*Elements_align*); un método para modificar el tamaño de los elementos (*Elements_size*).

Estos métodos modificarán el valor de los atributos que almacenan la distribución de los elementos en la interfaz (atributos ya existentes): *Window_position* almacena la posición de la interfaz en la pantalla; *Groups* almacena los grupos de campos de las UI de Servicio; *WidgetX_position* almacena la posición del elemento X de la interfaz (campos de entrada, acciones, navegaciones, filtros, variables de filtro, criterios de ordenación, columnas); *WidgetX_align* almacena la alineación del elemento X de la interfaz (campos de entrada, variables de filtro); *WidgetX_size* almacena el tamaño del elemento X de la interfaz (las columnas de las UI de Población, el número de instancias precargadas, los campos de entrada, las variables de filtro). Todos estos atributos no añaden cambios al código generado porque ya existen actualmente.

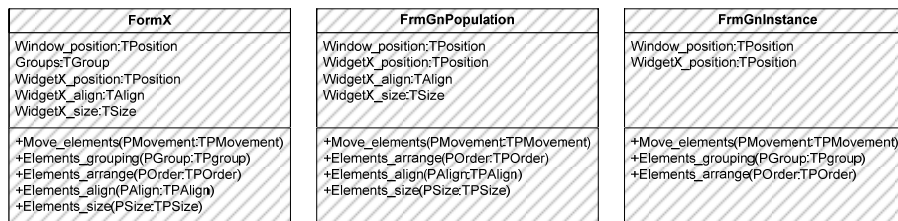


Figura 8.7 Diagrama de Clases para *Personal Object Space*

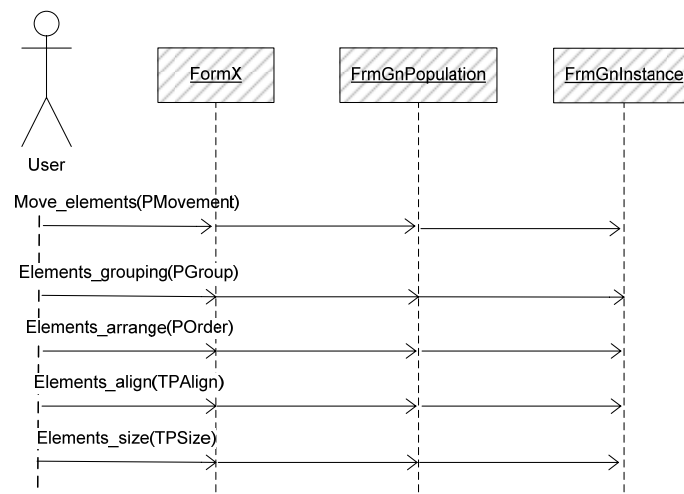


Figura 8.8 Diagrama de Secuencia para incorporar la forma de uso *Distribución de elementos*

El Diagrama de Secuencia que representa el mecanismo de usabilidad *Personal Object Space* (Figura 8.8) muestra que no hay un orden estricto en la invocación de los métodos de las clases. El usuario puede invocar indistintamente a los métodos de cualquiera de las tres clases modificadas por la incorporación del mecanismo de usabilidad.

La Tabla 8.12 muestra un resumen de los cambios que incorpora WU_POS1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	<ul style="list-style-type: none"> – Método para personalizar la posición donde se abre la interfaz dentro de la pantalla – Método para que el usuario agrupe campos de entrada – Método para personalizar la localización de campos de entrada – Método para personalizar la alineación de los campos de entrada – Método para personalizar el tamaño de los campos de entrada
FrmGnPopulation	<ul style="list-style-type: none"> – Método para personalizar la posición donde se abre la interfaz dentro de la pantalla – Método para personalizar el orden de los botones de acción – Método para personalizar el orden de los botones de Navegación – Método para personalizar el orden de las variables de filtro – Método para personalizar el orden de los filtros – Método para personalizar el orden de los criterios de ordenación – Método para personalizar el orden de las columnas

Clase Genérica	Cambios que Incorpora
	<ul style="list-style-type: none"> – Método para personalizar la alineación de las variables de filtro – Método para personalizar el tamaño de las columnas – Método para personalizar el número de instancias precargadas – Método para personalizar el tamaño de las variables de filtro
FrmGnInstance	<ul style="list-style-type: none"> – Método para personalizar la posición donde se abre la interfaz dentro de la pantalla – Método para personalizar el orden de los botones de acción – Método para personalizar el orden de los botones de Navegación – Método para personalizar el orden de las columnas – Método para personalizar el tamaño de las columnas

Tabla 8.12 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Distribución de elementos*

8.2.4.2 Resumen de los cambios que provoca Personal Object Space

La Tabla 8.13 muestra un resumen de la forma de uso del mecanismo de usabilidad *Personal Object Space*, las propiedades que la componen, los modelos conceptuales que se ven afectados y en qué manera, y los cambios a realizar en el compilador de modelos para implementar el mecanismo de usabilidad.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Distribución de elementos	Mover elementos	Modelo de Interacción Concreto	Habilitar que el usuario pueda modificar la posición de los elementos de la interfaz	<i>Form</i> X, <i>FrmGnPopulation</i> y <i>FrmGnInstance</i> añaden métodos para modificar la posición de los elementos de la interfaz
	Agrupar elementos	Modelo de Interacción Concreto	Habilitar que el usuario pueda agrupar los elementos de la interfaz	<i>Form</i> X, <i>FrmGnPopulation</i> y <i>FrmGnInstance</i> añaden métodos para agrupar elementos de la interfaz
	Ordenar elementos	Modelo de Interacción Concreto	Habilitar que el usuario pueda ordenar los campos de entrada, las acciones, las navegaciones, las variables de filtro, los filtros, los criterios de ordenación	<i>Form</i> X, <i>FrmGnPopulation</i> y <i>FrmGnInstance</i> añaden métodos para ordenar los campos de entrada, las acciones, las navegaciones, las variables de filtro, los filtros, los criterios de ordenación
	Alinear elementos	Modelo de Interacción Concreto	Habilitar que el usuario alinee los campos de entrada y las variables de filtro	<i>Form</i> X y <i>FrmGnPopulation</i> añaden métodos para alinear campos de entrada y variables de filtro

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
	Tamaño de los elementos	Modelo de Interacción Concreto	Habilitar que el usuario modifique el tamaño de las columnas, los campos de entrada, las variables de filtro y el número de instancias	<i>Form</i> X, <i>FrmGnPopulation</i> y <i>FrmGnInstance</i> añaden métodos para modificar el tamaño de las columnas, los campos de entrada, las variables de filtro y el número de instancias

Tabla 8.13 Resumen de los cambios para incorporar *Personal Object Space*

8.3 Favourites

Este mecanismo de usabilidad se utiliza para crear accesos rápidos a escenarios, de forma que el usuario pueda acceder fácilmente a aquellos que le sean de gran interés. Es decir, este mecanismo de usabilidad añade una nueva entrada al menú principal de la aplicación (menú de favoritos), donde el usuario puede añadir referencias a escenarios según sus preferencias. La aplicación de este mecanismo de usabilidad sobre el sistema incrementa la eficiencia del usuario, ya que puede encontrar los escenarios con los que trabaja habitualmente en menor tiempo. Este mecanismo tiene como objetivo el mismo que el patrón de usabilidad *Favoritos* de Welie [Welie, 2000].

8.3.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *Favourites* (Anexo I) se deriva una única forma de uso, **Definición de favoritos (WU_F1)**. La Tabla 8.14 muestra la pregunta de la guía de

captura de requisitos de la que se deriva esta forma de uso y el objetivo que se pretende conseguir con ella.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Permitirá el sistema almacenar los distintos contextos de interés que visite el usuario?	Permitir al usuario agregar al menú de favoritos los contextos más utilizados	Definición de favoritos (WU_F1)

Tabla 8.14 Derivación de la forma de uso de *Favourites* a partir de la guía de captura de requisitos

La pregunta de la guía de captura de requisitos de la que se deriva esta forma de uso es *¿Permitirá el sistema almacenar los distintos contextos de interés que visite el usuario?* El objetivo de esta forma de uso es dotar al sistema de la capacidad de almacenar los contextos (interfaces) que son de mayor interés para el usuario en una lista de favoritos. De esta manera, el usuario podrá acceder de forma rápida a ellos a través de un menú de favoritos.

Por ejemplo, en el sistema de alquiler de coches, la acción para dar de alta un cliente se accede desde la interfaz en la que se reserva el alquiler de un coche en caso de que el cliente no esté dado de alta. Una forma de mejorar la usabilidad del sistema sería que el usuario pudiera añadir al menú de favoritos una entrada que abriera directamente la interfaz para dar de alta un cliente sin necesidad de tener que entrar a la interfaz de realizar un alquiler.

8.3.2 Propiedades

Esta sección se centra en explicar las **propiedades de Definición de favoritos (WU_F1)**. Esta forma de uso tiene dos propiedades mediante las cuales el analista puede habilitar al usuario para que agregue contextos al menú de favoritos y decida dónde se va a visualizar la lista de favoritos. La Tabla 8.15 muestra una visión global de las preguntas de la guía de captura

de requisitos de la que se han derivado las propiedades y los objetivos que se pretenden conseguir con ella.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Permitirá el sistema almacenar los distintos contextos de interés que visite el usuario?	Habilitar que el usuario pueda añadir entradas al menú de favoritos	Habilitar favoritos (P1_WU_F1)	Configurable
¿Cómo se almacenará la lista de favoritos?	Seleccionar cómo se accederá a la lista de favoritos	Forma de acceso (P1_WU_F1)	Configurable

Tabla 8.15 Derivación de las propiedad de la forma de uso *Definición de favoritos* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades de WU_F1 son las siguientes:

- **Habilitar favoritos (P1_WU_F1):** La pregunta de la guía de captura de requisitos de la que se deriva la propiedad *Habilitar favoritos* es *¿Permitirá el sistema almacenar los distintos contextos de interés que visite el usuario?* El objetivo de esta propiedad es que el analista habilite o no la posibilidad de que el usuario añada entradas al menú de favoritos. Cada una de las entradas de dicho menú proporciona una forma rápida de acceder a los contextos más utilizados por el usuario. Esta propiedad es configurable, ya que es el analista el que debe especificar su valor.

Si el analista habilitara esta propiedad, el usuario debería especificar la siguiente información para configurar cada nueva entrada del menú de favoritos:

- **Contexto al que se accede:** Esta información que debe proporcionar el usuario aparece en la guía de captura de requisitos con la pregunta *¿Cuántos contextos se van a almacenar?* De esta manera el usuario selecciona los contextos que formarán parte del menú de favoritos. La lista de favoritos debe ser dinámica, es decir, el usuario puede introducir y eliminar elementos en cualquier momento.
- **Etiqueta:** Esta información que debe proporcionar el usuario aparece en la guía de captura de requisitos con la pregunta *¿Cómo se van a almacenar los contextos?* De esta manera, el usuario proporciona un alias a cada una de las entradas del menú de favoritos que defina. Si no introdujera ninguna etiqueta, la entrada del menú tomaría el alias del contexto.
- **Ordenación del menú:** Esta información que debe proporcionar el usuario aparece en la guía de captura de requisitos con la pregunta *¿Cómo se van a almacenar los contextos?* De esta forma el usuario puede definir su propio criterio de ordenación para aplicarlo a las entradas del menú de favoritos.
- **Forma de acceso (P2_WU_F1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Cómo se almacenará la lista de favoritos?* El objetivo de esta propiedad es el de definir en qué parte del sistema se mostrará la lista de favoritos. El analista debería decidir en qué entrada del menú de la aplicación introduce esta funcionalidad y si será accesible mediante teclas de acceso rápido (Por ejemplo, CTRL+F). Esta propiedad es configurable.

Por ejemplo, en el sistema de alquiler de coches, el analista podría habilitar que el usuario introdujera entradas al menú de favoritos mediante la propiedad *Habilitar favoritos (P1_WU_F1)*. De esta manera, el usuario puede añadir al menú de favoritos una entrada para acceder al servicio de alta de cliente sin necesidad de entrar a la interfaz de reservar alquiler de coche (único contexto desde el que se podía abrir el contexto alta de cliente). La Figura 8.9 representa la interfaz alta de cliente pero con la propiedad *Habilitar favoritos (P1_WU_F1)* activada. Una vez el usuario

pulsara sobre el botón de *Favoritos* (botón marcado en rojo), se mostraría una ventana (Figura 8.10) en la que debería introducir una etiqueta para la entrada del menú que lleve a ese contexto y el orden que tomaría el nuevo contexto en la lista de favoritos. Desde una ventana de gestión de favoritos, el usuario podría eliminar elementos del menú, modificar su orden y modificar sus etiquetas.

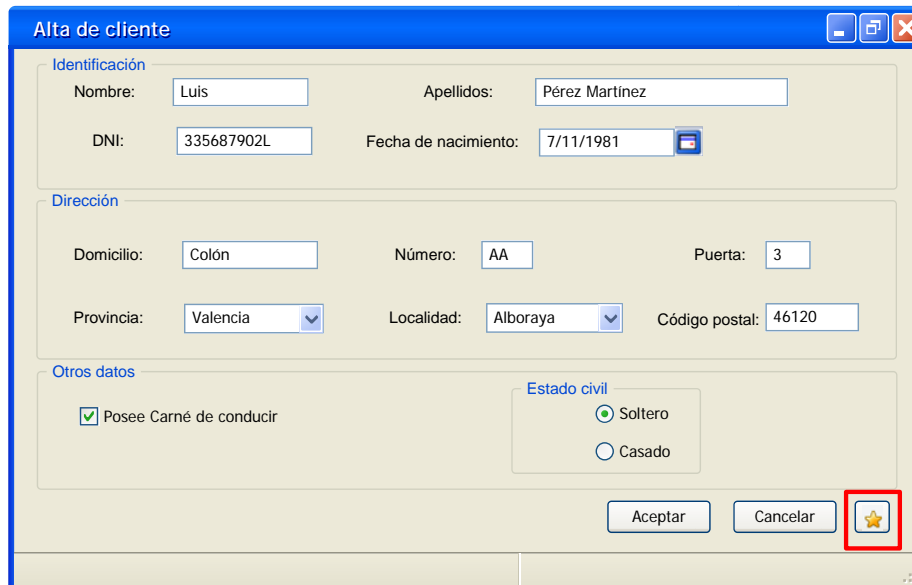


Figura 8.9 Alta de cliente con la forma de uso *Definición de favoritos*

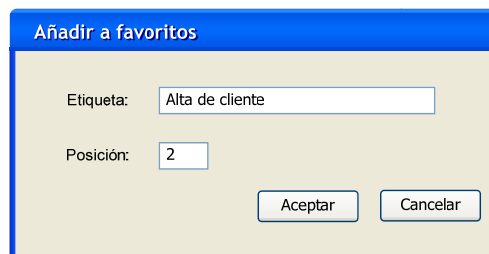


Figura 8.10 Alta de un nuevo favorito

La propiedad *Forma de acceso* (*P2_WU_F1*), tomaría el valor de una entrada en el menú del sistema, llamada *Favoritos*. La Figura 8.11 muestra

el menú del sistema donde se encuentra el acceso a la lista de favoritos (menú favoritos).

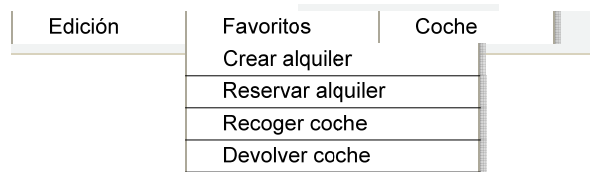


Figura 8.11 Menú con la lista de favoritos

La siguiente sección muestra cómo las propiedades configurables de *Definición de favoritos* afecta al modelado conceptual de OO-Method.

8.3.3 Modificación de los modelos conceptuales de OO-Method

Las aplicaciones generadas con OO-Method no disponen de ninguna funcionalidad para añadir elementos al menú de favoritos. Los cambios que se deben incorporar a OO-Method para soportar la forma de uso **Definición de favoritos (WU_F1)** afectan tanto al modelado por parte del analista como a la interacción del sistema con el usuario. Por un lado, el analista debe tener la capacidad de habilitar o no al usuario para que añada contextos al menú de favoritos, ya que dependiendo del dominio de la aplicación, el menú de favoritos puede ayudar o entorpecer la labor del usuario. Además, el analista tener la capacidad de decidir dónde se va a visualizar la lista de favoritos. Por otro lado, el usuario debe poder hacer todas estas acciones:

- Introducir una etiqueta para cada una de las entradas del menú de favoritos. En caso de que no introdujera ninguna etiqueta, se tomaría como valor por defecto el alias del contexto (interfaz).
- Indicar el orden de los elementos de la lista de favoritos.
- Modificar los elementos de la lista de favoritos (agregar, eliminar y modificar orden y etiquetas).

La Tabla 8.16 muestra la relación entre los cambios que se deben incorporar a OO-Method y las propiedades de las que se derivan. En la siguiente sección se muestra en detalle las nuevas primitivas conceptuales para incorporar esta propiedad a OO-Method.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Definición de favoritos (WU_F1)	Habilitar favoritos (P1_WU_F1)	Configurable	<ul style="list-style-type: none"> – El analista debe poder habilitar la gestión de favoritos al usuario – El usuario debe poder añadir y eliminar favoritos – El usuario debe poder definir una etiqueta para cada entra del menú favoritos – El usuario debe poder definir el orden de la lista de favoritos – El usuario debe poder modificar las características de los elementos de la lista de favoritos
	Forma de acceso (P2_WU_F1)	Configurable	El analista debe elegir cómo se accede a la lista de favoritos

Tabla 8.16 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Favourites*

8.3.3.1 WU_F1: Definición de favoritos

Esta sección explica los cambios que son necesarios llevar a cabo en el modelo conceptual de OO-Method para incorporar la forma de uso *Definir un asistente (WU_F1)*. Por lo que respecta a la propiedad *Habilitar favoritos*, está relacionada con el cómo se visualizará la información (el menú de favoritos), por lo tanto es razonables que esta propiedad se modele dentro del **Modelo de Interacción Concreto**. Este modelo debe añadir una nueva primitiva conceptual mediante la cual el analista pueda habilitar al usuario la gestión de los favoritos. Por defecto, la gestión de favoritos estará habilitada.

En cuanto a la otra propiedad configurable de WU_F1, *Ubicación de la lista de favoritos*, se modela dentro del **Modelo de Interacción Abstracto**, ya que afecta al Árbol de Jerarquía de Acciones que se modela dentro de dicho modelo. Se deben añadir dos nuevas primitivas conceptuales, una que indique con qué entrada del menú del sistema se accederá y otra que indique con qué secuencia de teclas. Por defecto, la lista de favoritos se mostrará desde la entrada del menú *Favoritos* y con las teclas *CTRL+F*. La Tabla 8.17 muestra de forma resumida los cambios que incorporan a nivel conceptual la forma de uso *Definición de favoritos*.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Concreto	Habilitar al usuario para gestione la lista de favoritos
Modelo de Interacción Abstracto	<ul style="list-style-type: none"> – Decidir la entrada del menú del sistema para acceder a la lista de favoritos – Decidir la combinación de teclas de acceso rápido para acceder a la lista de favoritos

Tabla 8.17 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Definición de favoritos*

En la siguiente sección se detallan los cambios que se deben realizar en el compilador de modelos para dar soporte a la nueva primitiva conceptual.

8.3.4 Modificación del compilador de modelos

8.3.4.1 WU_F1: Definición de favoritos

Para permitir que el usuario pueda personalizar la entrada del menú de favoritos en la aplicación es necesario introducir cambios en el compilador de modelos. Una vez el analista haya habilitado la posibilidad de que el usuario gestione la lista de favoritos, hay generar el código que proporcione soporte a dicha gestión.

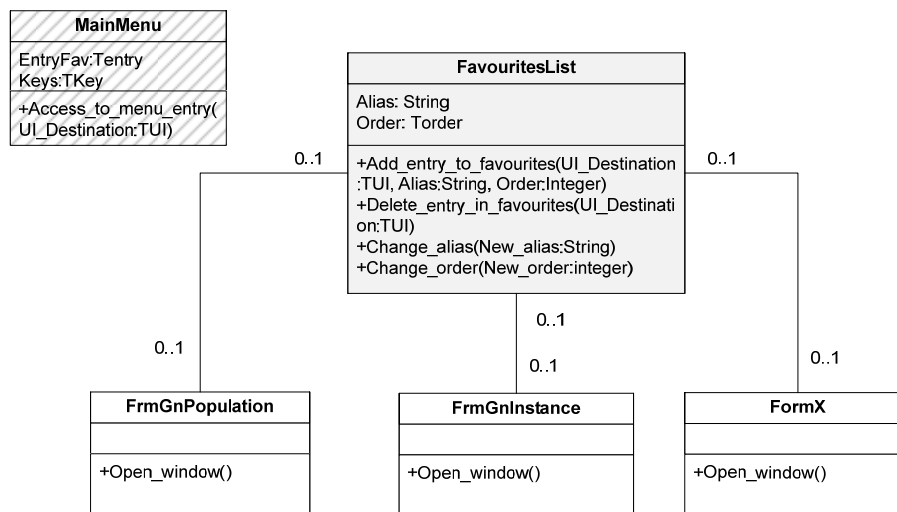


Figura 8.12 Diagrama de Clases para *Favourites*

La Figura 8.12 muestra en un diagrama de clases los cambios necesarios para implementar WU_F1. Se debe añadir una nueva clase llamada *FavouritesList* que da soporte a la lista de favoritos. Esta clase incluye: un método para añadir elementos al menú de favoritos, *Add_entry_to_favourites*; un método para borrarlos, *Delete_entry_in_favourites*; un método para modificar la etiqueta de una entrada de la lista de favoritos, *Change_alias*; un método para modificar el orden de un elemento en la lista *Change_order*. Todos estos métodos sólo se generarán en el código si el analista habilita los favoritos desde el modelo conceptual. Una vez habilitado, el usuario podrá hacer uso de estos

métodos para gestionar la lista de favoritos. La propiedad *Forma de acceso* se implementa en la clase *MainMenu*, donde actualmente ya se especifica el menú del sistema. Basta con añadir una nueva entrada con el menú de favoritos, pero no es necesario añadir nuevos atributos ni métodos. En el atributo *EntryFav* se almacena el contexto al cual se navegará y en el atributo *Keys* la combinación de teclas de acceso rápido que navegan hacia el contexto. La navegación se produce gracias al método *Access_to_menu_entry*, que navegará hacia el contexto favorito (UIP, UII o UIS).

El Diagrama de Secuencia de la Figura 8.13 muestra la secuencia de llamadas entre las clases que participan en la implementación de la forma de uso *Definición de favoritos (WU_F1)*. En este diagrama, el usuario introduce una nueva entrada en el menú de favoritos (mediante el método *Add_entry_to_favourites*) y posteriormente accederá a ella (mediante el método *Access_to_menu_entry*). La nueva entrada del menú de favoritos puede abrir una UIP, una UII o una UIS, dependiendo de cómo haya definido el usuario dicha entrada del menú.

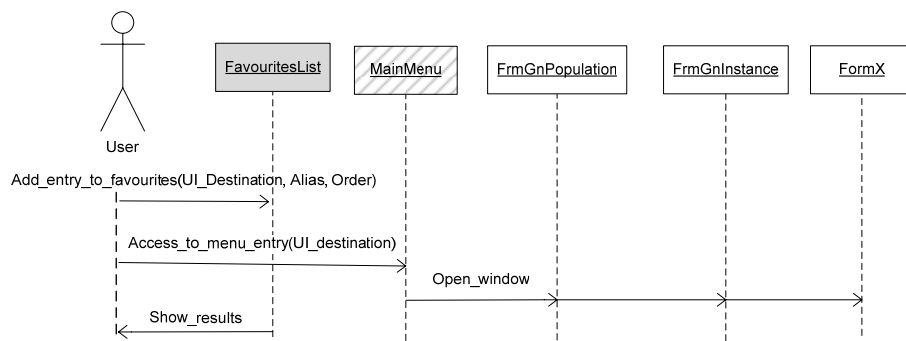


Figura 8.13 Diagrama de Secuencia para incorporar la forma de uso *Definición de favoritos*

La Tabla 8.18 muestra un resumen de los cambios que incorpora WU_POS1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
FavouritesList	<ul style="list-style-type: none"> – Método para agregar elementos a la lista de favoritos – Método para eliminar elementos de la lista de favoritos – Método para modificar el alias de la entrada del menú de favoritos – Método para modificar el orden de la entrada de la lista del menú de favoritos

Tabla 8.18 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Definición de Favoritos*

8.3.4.2 Resumen de los cambios que provoca Favourites

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Definición de favoritos	Habilitar favoritos	Modelo de Interacción Concreto	Habilitar al usuario para gestionar el menú de favoritos	<i>FavouritesList</i> debe permitir al usuario agregar, eliminar y modificar entradas del menú de favoritos
	Forma de acceso	Modelo de Interacción Abstracto	Se debe poder decidir cómo acceder a la lista de favoritos	<i>MainMenu</i> almacena la forma de acceso a la lista de favoritos

Tabla 8.19 Resumen de los cambios para incorporar *Favourites*

La Tabla 8.19 muestra un resumen de la forma de uso del mecanismo de usabilidad *Favourites*, la propiedad que la compone, los modelos conceptuales que se ven afectados, las nuevas primitivas conceptuales necesarias para su representación y los cambios a realizar en el compilador de modelos para implementar el mecanismo de usabilidad.

Capítulo 9

Incorporación de Undo y Cancel

Undo y Cancel es un FUF que tiene como objetivo permitir al usuario cancelar una acción o volver a un estado anterior al que se encuentra. A continuación se presenta una sección por cada mecanismo de usabilidad en los que se divide: *Global Undo* y *Abort Operation*. En cada sección se aplica el método MIMAT a un mecanismo de usabilidad distinto.

9.1 Global Undo

Este mecanismo de usabilidad incorpora a los sistemas la posibilidad de deshacer las últimas acciones que haya realizado el usuario. Los usuarios suelen explorar los sistemas sin tener conocimientos sobre el mismo, y por lo tanto suelen cometer errores. Mediante este mecanismo se ayuda a resolver los problemas que hayan surgido durante esta exploración, ya que convierte los errores de irrevocables a revocables. Este mecanismo de usabilidad tiene el mismo objetivo que el patrón de usabilidad *Ir un paso atrás* de Tidwell [Tidwell, 2005].

9.1.1 Formas de uso

De la guía de captura de requisitos del mecanismo de usabilidad *Global Undo* (Anexo I) se obtienen las siguientes formas de uso:

1. WU_GU1: Deshacer cambios.
2. WU_GU2: Rehacer cambios.

La Tabla 9.1 muestra una visión global de las formas de uso, las preguntas de la guía de captura de requisitos a partir de la cuales se han derivado y los objetivos que se pretenden conseguir con cada una de ellas.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Qué acciones serán permanentes y cuáles se podrán deshacer?	Deshacer las acciones ejecutadas por error	Deshacer cambios (WU_GU1)
¿Requerirá el sistema diferentes combinaciones de deshacer y rehacer o sólo la funcionalidad de deshacer?	Rehacer acciones que previamente se han deshecho	Rehacer cambios (WU_GU2)

Tabla 9.1 Derivación de las formas de uso de *Global Undo* a partir de la guía de captura de requisitos

La primera de las formas de uso, **Deshacer cambios (WU_GU1)**, se deriva de la pregunta de la guía de captura de requisitos, *¿Qué acciones serán permanentes y cuáles se podrán deshacer?* El objetivo de esta forma de uso es dotar al sistema de la capacidad de deshacer las acciones que haya ejecutado el usuario de forma errónea. Sería conveniente dotar al sistema con la funcionalidad de esta forma de uso siempre que una acción se pueda deshacer en base a la lógica de negocio. Esto corregiría de manera sencilla un gran número de errores cometidos por el usuario. Los cambios registrados en el sistema se deberían almacenar en una especie de pila, donde cada operación de deshacer corresponde a un elemento de la pila y el último cambio realizado en el sistema es el primer elemento de dicha pila.

Por ejemplo, en el sistema de alquiler de coches, el servicio de poner un coche en venta debería poder ser deshecho, ya que el administrador del sistema se puede equivocar a la hora de seleccionar la matrícula del coche a poner en venta.

La segunda forma de uso, **Rehacer cambios (WU_GU2)**, se deriva de la pregunta de la guía de captura de requisitos *¿Requerirá el sistema diferentes combinaciones de deshacer y rehacer o sólo la funcionalidad de deshacer?* El objetivo de esta forma de uso es el de dotar al sistema de la capacidad de volver a hacer las acciones que el usuario haya deshecho previamente. Al igual que WU_GU1, esta otra forma de uso almacena las acciones que se pueden rehacer en una pila.

Por ejemplo, en el sistema de alquiler de coches, una vez el usuario haya deshecho el servicio poner coche en venta, podría volver a rehacerlo gracias a esta forma de uso. En la siguiente sección se van a explicar las propiedades que componen cada una de estas formas de uso.

9.1.2 Propiedades

Por un lado, las **propiedades de Deshacer cambios (WU_GU1)** marcan cuáles serán los servicios y los cambios de interfaz con capacidad para deshacerse, además de determinar la forma en la que se presentará esta funcionalidad al usuario. La Tabla 9.2 muestra de manera resumida cómo se han derivado las propiedades a partir de las preguntas de la guía de captura de requisitos.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué acciones serán permanentes y cuáles se podrán deshacer?	Indicar los servicios con la capacidad de deshacer los resultados de su ejecución	Selección de servicios (P1_WU_GU1)	Configurable
¿Qué acciones serán permanentes y cuáles se podrán deshacer?	Indicar qué elementos de la interfaz podrán deshacer las modificaciones que haya hecho el usuario	Selección de elementos de la interfaz (P2_WU_GU1)	Configurable

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Cuántos niveles de deshacer son necesarios?	Indicar el número máximo de acciones que se pueden deshacer	Número máximo de niveles para deshacer (P3_WU_GU1)	No configurable
¿Cuál es la mejor forma de presentar al usuario la pila de elementos que se pueden deshacer?	Determinar cómo podrá el usuario deshacer sus acciones con el sistema	Forma de acceso (P4_WU_GU1)	Configurable

Tabla 9.2 Derivación de propiedades de la forma de uso *Deshacer cambios* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades son las siguientes:

- **Selección de servicios (P1_WU_GU1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué acciones serán permanentes y cuáles se podrán deshacer?* El objetivo de esta propiedad es que el analista especifique los servicios del sistema que tendrán la capacidad de poder deshacer las consecuencias de su ejecución, es decir, dejar la base de datos tal y como estaba antes de la ejecución del servicio. Dependiendo de la naturaleza del servicio, la operación de deshacer tendrá sentido o no. Por ejemplo, en el sistema de alquiler de coches no sería correcto permitir deshacer el servicio pagar alquiler una vez se le ha cobrado al cliente, ya que aunque se deshaga la acción de la base de datos el cobro ya se habrá efectuado. Esta propiedad es por tanto configurable.

- **Selección de elementos de la interfaz (P2_WU_GU1):** Esta propiedad también se deriva de la pregunta de la guía de captura de requisitos, *¿Qué acciones serán permanentes y cuáles se podrán deshacer?* El objetivo de esta propiedad es determinar qué elementos de la interfaz van a tener la capacidad de deshacer las modificaciones que haga el usuario en ellos. Los elementos de la interfaz que pueden tener la capacidad de deshacer son: campos de entrada de datos, modificación de imágenes, cambios en la presentación del contenido, cualquier operación de cortado y pegado. Esta propiedad es configurable.
- **Número máximo de niveles para deshacer (P3_WU_GU1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Cuántos niveles de deshacer son necesarios?* El objetivo de esta propiedad es el de determinar el tamaño de la pila en la que se irán almacenando los servicios con la capacidad de deshacer. Siguiendo las recomendaciones de [Tidwell, 2005], se deberían almacenar los últimos diez servicios ejecutados por el usuario, por lo tanto, éste será el número de elementos que se podrán almacenar en la pila. Esta propiedad es no configurable, ya que su valor viene dado por el patrón de usabilidad de Tidwell y no por el analista.
- **Forma de acceso (P4_WU_GU1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Cuál es la mejor forma de presentar al usuario la pila de elementos que se pueden deshacer?* El objetivo de esta propiedad es definir cómo se presentará al usuario la posibilidad de deshacer servicios. El analista debería decidir en qué entrada del menú de la aplicación introduce esta funcionalidad y si será accesible mediante teclas de acceso rápido (Por ejemplo, CTRL+Z). Por lo tanto, es una propiedad configurable.

Por ejemplo, en el sistema de alquiler de coches hay un servicio (poner en venta) para poner en venta aquellos coches que ya no se van a utilizar para alquilar. La interfaz de la Figura 9.1 representa una posible implementación de ese servicio. En este caso, sería conveniente que el analista incorporara al servicio poner en venta la capacidad de deshacerse. Para ello, la

propiedad *Selección de servicios (P1_WU_GU1)* tomaría el valor poner en venta. De esta manera, si el usuario se equivocara de matrícula, se podría deshacer la acción y el coche dejaría de estar en venta para pasar a estar disponible para alquilar.

Figura 9.1 Poner coche en venta con la capacidad de deshacer

Matrícula	Marca	Modelo	Caballos	Motor	Color	Precio
4587 FVR	SEAT	Ibiza	80	Gasolina	Rojo	12.000
9297 BFH	SEAT	Ibiza	120	Diesel	Negro	10.000
4994 DPG	SEAT	León	105	Gasolina	Blanco	15.000
8861 CPF	FORD	Focus	120	Diesel	Plata	10.300
8811 FGT	FORD	Fiesta	80	Diesel	Rojo	10.000
1957 FWT	CITROEN	C3	80	Gasolina	Azul	9.000

Figura 9.2 Listado de coches en venta

Por ejemplo, si la última acción ha sido poner en venta el coche con matrícula 4587 FVR, desde la ventana de listado de coches en venta (Figura 9.2) el usuario podría deshacer dicha acción. Para este ejemplo, la propiedad *Selección de elementos de la interfaz (P2_WU_GU1)* tomaría el valor de todos los campos de entrada de la interfaz. Así el usuario también podría deshacer cualquier valor introducido en los campos antes de que

pulsara el botón *Aceptar*. La propiedad *Número máximo de niveles para deshacer (P3_WU_GU1)* tomaría el valor diez y mediante la propiedad *Forma de acceso (P4_WU_GU1)* se indicaría que el usuario puede invocar a deshacer desde el menú *Edición* o con *CTRL+Z*.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué acciones serán permanentes y cuáles se podrán rehacer?	Indicar los servicios con la capacidad de deshacer los resultados de su ejecución	Selección de servicios (P1_WU_GU2)	Configurable
¿Qué acciones serán permanentes y cuáles se podrán rehacer?	Indicar qué elementos de la interfaz podrán deshacer las modificaciones que haya hecho el usuario	Selección de elementos de la interfaz (P2_WU_GU2)	Configurable
¿Cuántos niveles de rehacer son necesarios?	Indicar el número máximo de acciones que se pueden deshacer	Número máximo de niveles para rehacer (P3_WU_GU2)	No configurable
¿Cuál es la mejor forma de presentar al usuario la pila de elementos que se pueden rehacer?	Determinar cómo podrá el usuario deshacer sus acciones con el sistema	Forma de acceso (P4_WU_GU2)	Configurable

Tabla 9.3 Derivación de propiedades de la forma de uso *Rehacer cambios* a partir de la guía de captura de requisitos

Por otro lado, las **propiedades de Rehacer cambios (WU_GU2)** se utilizan para determinar sobre qué servicios y elementos de la interfaz se pueden rehacer acciones y en qué forma se presentará esta funcionalidad al usuario. La Tabla 9.3 presenta las preguntas de la guía de captura de requisitos de la que se han derivado las propiedades y el objetivo de cada una de esas propiedades.

Tal como se puede apreciar, las propiedades de WU_GU2 son las mismas que para WU_GU1. Lo único que varía es que en vez de configurar la acción deshacer, lo que se configura con ellas es la acción rehacer. Como ejemplo para ver la aplicación de las propiedades, se ha seleccionado el servicio poner en venta. Una vez el usuario hubiera deshecho la ejecución del servicio poner en venta para el coche con matrícula *4587 FVR*, el usuario podría volver a ponerlo en venta si el analista hubiera introducido el servicio poner en venta dentro de la propiedad *Selección de servicios (P1_WU_GU2)*. La propiedad *Selección de elementos de la interfaz (P2_WU_GU2)* tomaría como valor todos los campos de entrada de la interfaz. De esta manera se podrá rehacer las modificaciones deshechas sobre dichos campos. La propiedad *Número máximo de niveles para rehacer (P3_WU_GU2)* tomaría el valor diez y el valor de la propiedad *Forma de acceso (P4_WU_GU2)* indicaría que el usuario puede hacer uso de la acción deshacer desde el menú *Edición* o con las teclas *CTRL+Y*.

9.1.3 Modificación de los modelos conceptuales de OO-Method

De las propiedades de las dos formas de uso en las que se divide el mecanismo de usabilidad *Global undo*, la única que está soportada en parte es la de *Selección de elementos de la interfaz (P2_WU_GU1)* dentro de la forma de uso *Deshacer cambios*. Actualmente, los campos de entrada de las interfaces derivadas de las UISs son capaces de deshacer el último valor que se ha introducido en ellos, siempre que el argumento no sea de tipo objeto valuado. Una de las limitaciones del soporte que proporciona OO-Method a este mecanismo de usabilidad es que la propiedad *Número máximo de niveles para deshacer (P3_WU_GU1)* tiene únicamente el valor uno, y por tanto el sistema sólo puede deshacer al valor inmediatamente anterior. Otras limitaciones de la forma de uso *Deshacer acciones* son que

el sistema no permite deshacer la ejecución de ningún servicio ni el analista puede decidir cómo presentar la capacidad de deshacer al usuario.

Todas las propiedades de *Deshacer cambios* son interesantes para las aplicaciones generadas por OO-Method. La única restricción se da en la propiedad *Selección de elementos de la interfaz (P2_WU_GU1)*, ya que para el tipo de sistemas desarrollados con OO-Method sólo tiene sentido dotar de la capacidad de deshacer a los campos de entrada de las interfaces derivadas de las UISs y a las personalizaciones de las interfaces una vez se incorporen éstas a los sistemas generados. Por lo tanto, se pueden deshacer las acciones incorporadas por el mecanismo de usabilidad *Personal Object Space* y *Preferences*.

La incorporación de todas las propiedades de la forma de uso **Deshacer cambios (WU_GU1)** implican los siguientes cambios en el modelado conceptual de OO-Method:

- El analista debe poder seleccionar aquellos servicios que tendrán la capacidad de deshacerse una vez ejecutados.
- La pila de acciones para deshacer debe tener en cuenta no sólo la ejecución de servicios, sino los valores que el usuario haya introducido en los campos de entrada de las interfaces derivadas de una UIS, además de las personalizaciones en la interfaz y la distribución de sus elementos. La capacidad de deshacer los valores de los campos de entrada, las personalizaciones y la distribución deben estar presente en todos los sistemas a pesar de derivar de la propiedad configurable *Selección de elementos de la interfaz*. Esto se debe a que dichas funcionalidades no suponen una carga extra para la ejecución del sistema, como puede suponer las acciones de deshacer servicios. Además, todos estos elementos son susceptibles de deshacer sus valores y no existen casos en los que la capacidad de deshacer no esté permitida. Por lo tanto, esta propiedad al aplicarla a OO-Method es una propiedad establecida.
- El usuario debe ser capaz de almacenar hasta un máximo de diez elementos (acciones, valores de campos de entrada y personalizaciones) en la pila de acciones para deshacer.

- El analista debe poder especificar cómo accederá el usuario a la funcionalidad de deshacer.

En cuanto a la forma de uso **Rehacer cambios (WU_GU2)**, no está actualmente soportada ninguna de sus propiedades por OO-Method. Los cambios que se deben realizar en OO-Method para incorporarla son similares a los cambios que introduce WU_GU1:

- El analista debe ser capaz de seleccionar los servicios que tendrán la capacidad de rehacer su ejecución después de que el usuario la haya deshecho.
- El usuario debe ser capaz de rehacer el valor que tenía un campo de entrada después de haberlo deshecho o rehacer las personalizaciones y las distribuciones de elementos de la interfaz. Esta característica, a pesar de derivar de una propiedad configurable, debe estar incluida en todos los campos de entrada ya que no supone una carga excesiva en la ejecución del sistema y no se pueden dar casos en que su incorporación sea un inconveniente para la lógica del negocio. Por lo tanto, al aplicar a OO-Method esta propiedad se convierte en una propiedad establecida.
- El usuario debe poder almacenar en la pila de acciones para rehacer hasta un máximo de diez elementos.
- El analista debe poder especificar cómo accederá el usuario a la funcionalidad de rehacer.

La Tabla 9.4 muestra de manera resumida los cambios que provoca en OO-Method cada una de las propiedades de *Global Undo*. En la siguiente sección se detallan los cambios a nivel de modelado conceptual que es necesario llevar a cabo para la incorporación de las formas de uso *Deshacer cambios (WU_GU1)* y *Rehacer cambios (WU_GU2)*.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Deshacer cambios (WU_GU1)	Selección de servicios (P1_WU_GU1)	Configurable	El analista debe seleccionar los servicios que tendrán la capacidad de deshacer sus acciones
	Selección de elementos de la interfaz (P2_WU_GU1)	Establecida (No configurable)	El usuario debe ser capaz de deshacer cualquier valor introducido en los campos de entrada, las personalizaciones y la distribución de elementos de la interfaz
	Número máximo de niveles para deshacer (P3_WU_GU1)	No configurable	El usuario debe ser capaz de deshacer un máximo de diez acciones
	Forma de acceso (P4_WU_GU1)	Configurable	El analista debe poder especificar cómo accederá el usuario a la funcionalidad de deshacer
Rehacer cambios (WU_GU2)	Selección de servicios (P1_WU_GU2)	Configurable	El analista debe seleccionar los servicios que tendrán la capacidad de rehacer sus acciones deshechas

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
	Selección de elementos de la interfaz (P2_WU_GU2)	Establecida (No configurable)	El usuario debe ser capaz de rehacer cualquier valor deshecho en los campos de entrada, en la personalización y en la distribución de elementos de la interfaz
	Número máximo de niveles para deshacer (P3_WU_GU2)	No configurable	El usuario debe ser capaz de rehacer un máximo de diez acciones
	Forma de acceso (P4_WU_GU2)	Configurable	El analista debe poder especificar cómo accederá el usuario a la funcionalidad de rehacer

Tabla 9.4 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Global Undo*

9.1.3.1 WU_GU1: Deshacer cambios

Las propiedades configurables de WU_GU1 implican cambios en el modelado conceptual de OO-Method. Tal y como se ha comentado anteriormente, la propiedad *Selección de elementos de la interfaz (P2_WU_GU1)* a pesar de definirse como configurable, se convierte en no configurable al aplicarla a OO-Method (propiedad establecida). Por lo tanto, esta propiedad no incorpora cambios a nivel conceptual. El resto de propiedades configurables, *Selección de servicios (P1_WU_GU1)* y *Forma de acceso (P4_WU_GU1)* afectan a dos modelos distintos. Por un lado,

Selección de servicios define una característica de los servicios definidos en el Modelo de Objetos. Tiene sentido que el analista decida habilitar la capacidad de deshacer el servicio en el momento de su definición. Por tanto, esta propiedad se representa con nuevas primitivas en el **Modelo de Objetos**. Sólo es necesaria una primitiva para cada servicio definido que indique si dicho servicio tiene la capacidad de deshacerse o no. Por defecto, ningún servicio tendrá la capacidad de deshacer sus acciones

Por lo que respecta a *Forma de acceso (P4_WU_GU1)*, configura cómo accederá el usuario a la acción de deshacer. El menú del sistema actualmente se representa en OO-Method en el Modelo de Interacción Abstracto. Por lo tanto, en el **Modelo de Interacción Abstracto** se deben añadir dos primitivas conceptuales nuevas para acceder a la funcionalidad de deshacer, una que indique con qué entrada del menú del sistema se accederá y otra que indique con qué secuencia de teclas. Por defecto, la acción deshacer se mostrará en la entrada del menú *Edición* y con las teclas *CTRL+Z*.

La Tabla 9.5 muestra de forma resumida los cambios que incorpora *Deshacer cambios (WU_GU1)* al modelado conceptual de OO-Method.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Objetos	Selección de servicios con capacidad de deshacer sus ejecuciones
Modelo de Interacción Abstracto	<ul style="list-style-type: none"> – Decidir la entrada del menú del sistema para acceder a la funcionalidad de deshacer – Decidir la combinación de teclas de acceso rápido para acceder a la funcionalidad de deshacer

Tabla 9.5 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Deshacer cambios*

9.1.3.2 WU_GU2: Rehacer cambios

La forma de uso *Rehacer cambios* incorpora a OO-Method cambios similares a los que introduce *Deshacer cambios*. La propiedad *Selección de elementos de la interfaz (P2_WU_GU2)* a pesar de ser configurable se convierte en no configurable al aplicarla a OO-Method (propiedad establecida). Por lo tanto, esta propiedad no implica cambios a nivel conceptual. Las únicas propiedades que introducen cambios son *Selección de servicios (P1_WU_GU2)* y *Forma de acceso (P4_WU_GU2)*.

Por un lado, *Selección de servicio* se debe modelar desde el **Modelo de Objetos**, ya que es el momento de la definición de los servicios donde el analista debe indicar la capacidad de rehacer las ejecuciones deshechas. Para que esta capacidad se pueda habilitar, es necesario que previamente se haya habilitado la capacidad de deshacer. Por defecto, los servicios tendrán la capacidad de deshacerse inhabilitada. Por otro lado, *Forma de acceso (P4_WU_GU2)* configura cómo acceder a la funcionalidad de rehacer. Esta configuración se realiza hoy en día en el Modelo de Interacción Abstracto de OO-Method. Por lo tanto, esta propiedad afecta al **Modelo de Interacción Abstracto** incluyendo dos nuevas primitivas conceptuales para indicar cómo acceder a la funcionalidad de rehacer: una para indicar con qué entrada del menú del sistema se accederá y otra para indicar con qué secuencia de teclas. Por defecto, la acción rehacer se mostrará en la entrada del menú *Edición* y con las teclas *CTRL+Y*.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Objetos	Selección de servicios con capacidad de rehacer ejecuciones deshechas
Modelo de Interacción Abstracto	<ul style="list-style-type: none"> – Decidir la entrada del menú del sistema para acceder a la funcionalidad de rehacer – Decidir la combinación de teclas de acceso rápido para acceder a la funcionalidad de rehacer

Tabla 9.6 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Rehacer cambios*

La Tabla 9.6 muestra de forma resumida los cambios que incorpora *Rehacer cambios (WU_GU2)* al modelado conceptual de OO-Method.

Una vez detectados los cambios que se deben realizar en los modelos conceptuales, el siguiente paso es detectar los cambios que se deben realizar en el compilador de modelos.

9.1.4 Modificación del compilador de modelos

9.1.4.1 WU_GU1: Deshacer cambios

Tanto las nuevas primitivas conceptuales de *Deshacer cambios* explicadas en la sección anterior como las propiedades no configurables implican cambios en el compilador de modelos. La Figura 9.3 presenta un Diagrama de Clases con las clases que se ven afectadas por la incorporación de WU_GU1 y WU_GU2. A continuación se detallan los cambios provocados por WU_GU1 y en la siguiente sección se explicarán los provocados por WU_GU2.

Las clases sobre las que se puede ejecutar la acción deshacer son *FormX*, *FrmGnPopulation* y *FrmGnInstance*. Para invocar la acción de deshacer se debe incluir un método *Undo* a cada una de estas clases. Además de modificar estas clases, es necesario añadir clases nuevas como *Undo pile* que tiene el objetivo de implementar la pila de acciones a deshacer. Esta pila puede contener los resultados de la ejecución de servicios, valores de campos de entrada, personalizaciones y distribución de elementos de la interfaz. En la clase *Changes in the data base* se almacenan los cambios que los servicios con la capacidad de deshacer provocan en la base de datos. Cada vez que el usuario introduzca un valor en los campos de entrada se almacenarán en la clase *Undoable inserted elements*; cada vez que personalice algún elemento de la interfaz se almacenará el cambio en la clase *Undoable preferentes*; y cada vez que modifique la distribución de elementos se almacenará en *Undoable personal object space*. La propiedad *Forma de acceso* se implementa en la clase *MainMenu*, donde se detalla la entrada del menú principal que accede a la funcionalidad de deshacer y la combinación de teclas que formarán el acceso rápido.

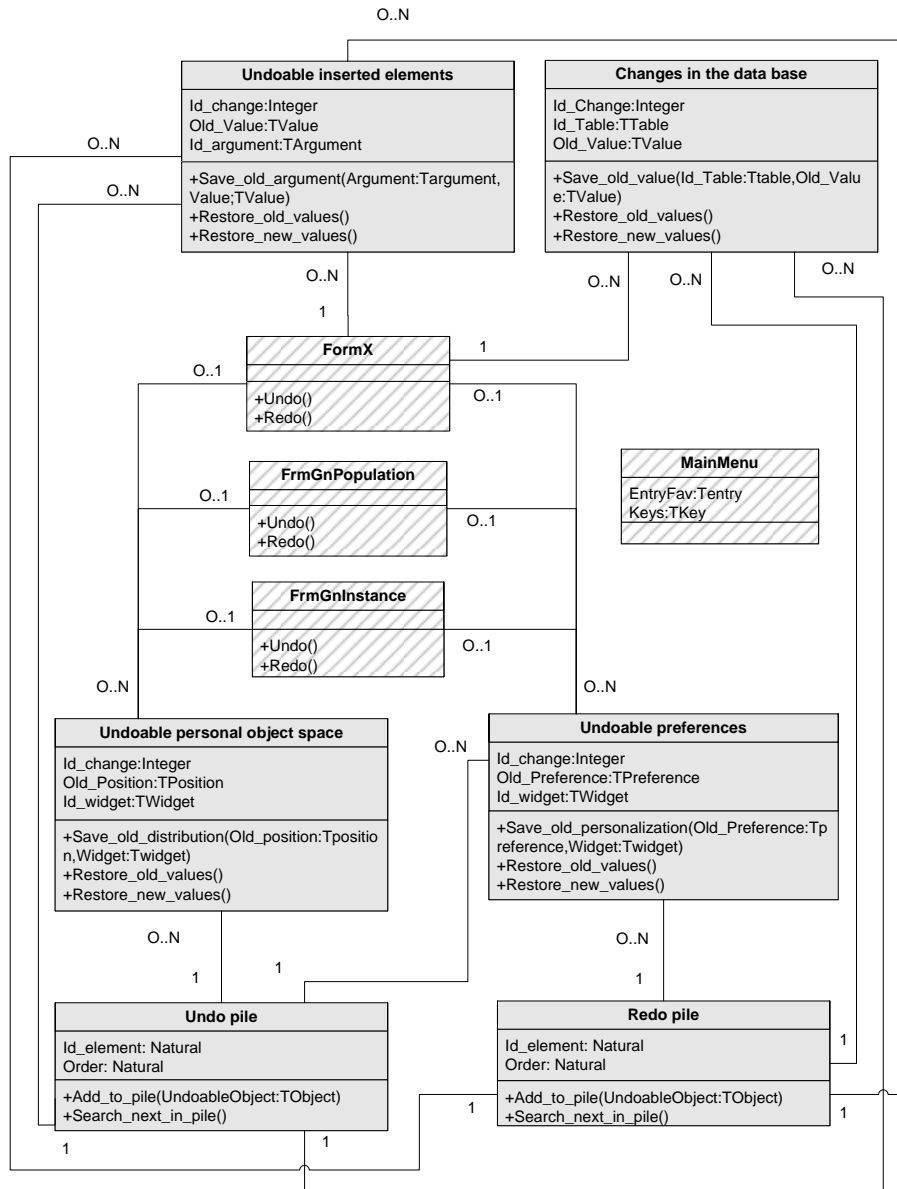


Figura 9.3 Diagrama de Clases para *Global Undo*

El diagrama de la Figura 9.4 representa la secuencia de llamadas que ocurrirán en cada modificación que haga el usuario sobre los campos de una UI de Servicio, o sobre la personalización o distribución de elementos de una UI de Servicio, de Población, o de Instancia. Además, también se

tendrán en cuenta las modificaciones en la base de datos provocadas por la ejecución de servicios. Las llamadas a los métodos *save_old_arguments* guardarán los valores previos a la modificación, y las llamadas a *add_to_pile* establecerán el orden de estas modificaciones. Mediante estas invocaciones, se almacenará de forma ordenada el histórico de modificaciones. La secuencia para almacenar en la pila los cambios que produce la ejecución de servicios será la misma, sólo que la clase en la que se almacenarán estos cambios es *Changes in the data base*.

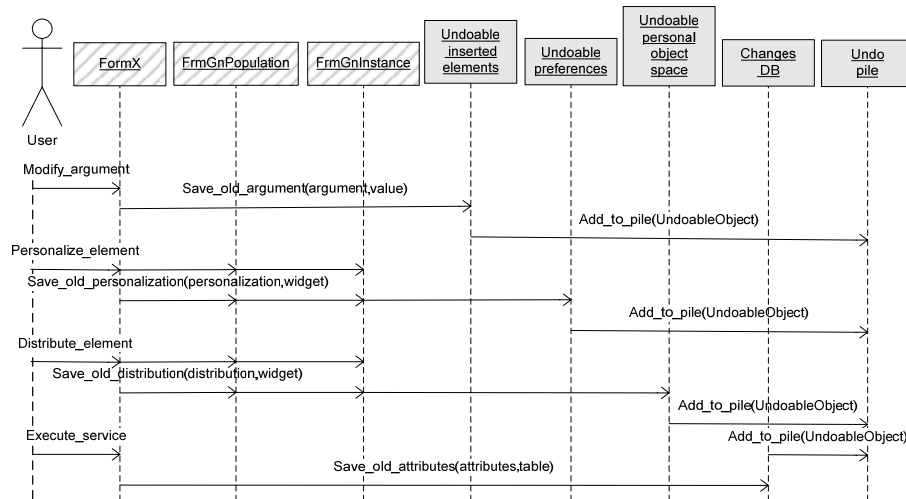


Figura 9.4 Diagrama de Secuencia para incorporar la forma de uso *Deshacer cambios* (almacenar antiguos valores para deshacer)

El Diagrama de Secuencia de la Figura 9.5 muestra la secuencia de invocaciones entre las clases que proporcionarán soporte a la acción de deshacer. La acción de rehacer se podrá invocar desde cualquiera de las tres clases que representan los tres tipos de UIs (*FormX*, *FrmGnPopulation*, *FrmGnInstance*). Una vez realizada esta llamada, se buscará cuál es el primer elemento disponible en la clase *Undo pile* y se cargarán los valores que ésta tenga almacenados. Estos valores pueden afectar al valor de campos de entrada, a la personalización de interfaces, a la disposición de elementos o a cambios de atributos en la base de datos. Los métodos *Change_personalization* y *Change_distribution* son una forma abstracta de representar el conjunto de métodos definidos en los Diagramas de Clases de la Figura 8.4 y Figura 8.7 respectivamente. Estos dos métodos engloban

todos los métodos que impliquen un cambio en la personalización o en la distribución de elementos.

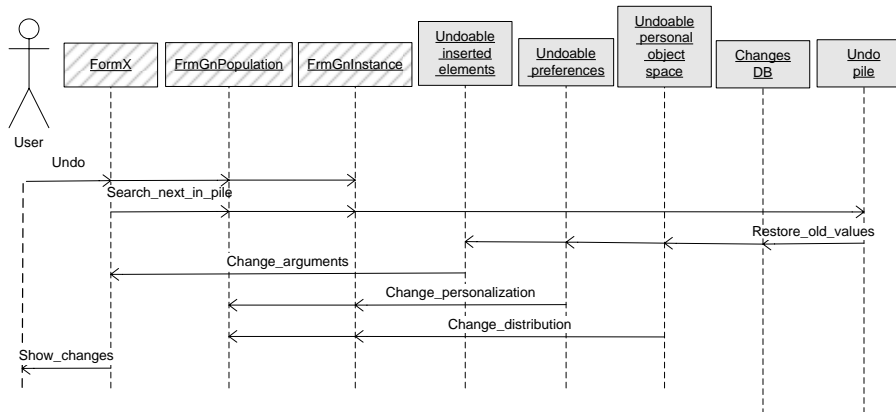


Figura 9.5 Diagrama de Secuencia para *Global Undo* (cargar antiguos valores al deshacer)

La Tabla 9.7 muestra un resumen de los cambios que incorpora WU_GU1 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	Tiene un método para iniciar la acción de deshacer
FrmGnPopulation	Tiene un método para iniciar la acción de deshacer
FrmGnInstance	Tiene un método para iniciar la acción de deshacer
Undoable inserted elements	<ul style="list-style-type: none"> - Almacena los valores introducidos en los campos de entrada - Extrae de la pila los valores anteriores de los campos de entrada
Undoable preferences	<ul style="list-style-type: none"> - Almacena las preferencias modificadas del usuario - Extrae de la pila las preferencias anteriores

Clase Genérica	Cambios que Incorpora
Undoable personal object space	<ul style="list-style-type: none"> – Almacena la distribución de los elementos seleccionada por el usuario – Extrae de la pila la anterior distribución
Changes in the data base	<ul style="list-style-type: none"> – Almacena los valores de los atributos de la base de datos antes de que la ejecución de un servicio los modifique – Extrae de la pila los valores de los atributos anteriores a la modificación
Undo pile	Almacena en forma de pila todos los cambios en campos de entrada, personalización, distribución de elementos y ejecución de servicios
MainMenu	Almacena la entrada del menú del sistema para acceder a la funcionalidad de deshacer y la combinación de teclas de acceso rápido

Tabla 9.7 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Deshacer cambios*

9.1.4.2 WU_GU2: Rehacer cambios

El Diagrama de Clases de la Figura 9.3 incluye las clases necesarias para rehacer los cambios que previamente se han deshecho. Los cambios que soporta la acción de rehacer son: valores de los campos de entrada (*Undoable inserted elements*), preferencias del usuario (*Undoable preferences*), distribución de elementos por la interfaz según los criterios del usuario (*Undoable personal object space*), valores de atributos modificados por la ejecución de servicios (*Changes in the data base*). Todas estas posibilidades de rehacer se almacenan en una pila mediante la clase *Redo pile*. La funcionalidad rehacer se accederá mediante la entrada del menú del sistema y las teclas de acceso rápido almacenadas en la clase *MainMenu*.

El Diagrama de Secuencia de la Figura 9.6 muestra la secuencia de llamadas para almacenar en la pila los valores que se podrán rehacer. Cada operación de *Undo* del usuario, implica guardar los valores que existan en ese momento antes de deshacerlos, por si el usuario quiere volver a

rehacerlos en un futuro. Estos valores una vez guardados se añadirán a la clase *Redo pile* para ordenarlos en forma de pila.

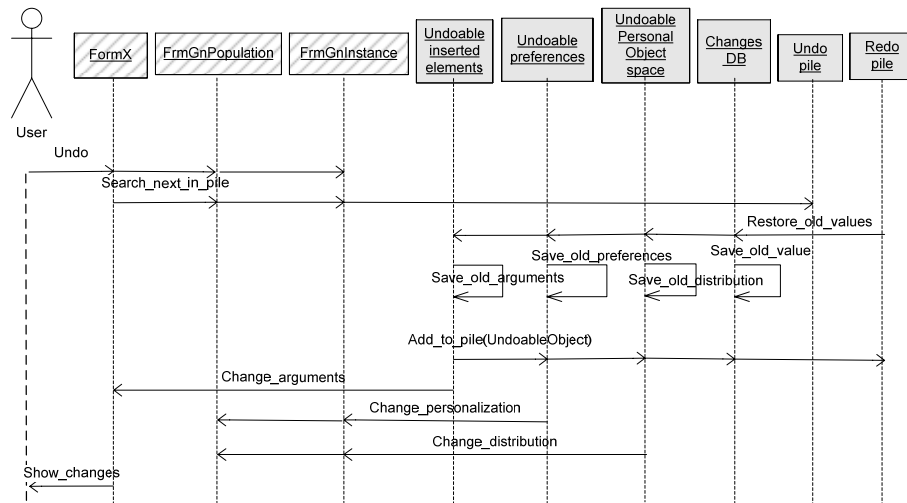


Figura 9.6 Diagrama de Secuencia para incorporar la forma de uso *Rehacer cambios* (cargar antiguos valores en la pila para rehacer)

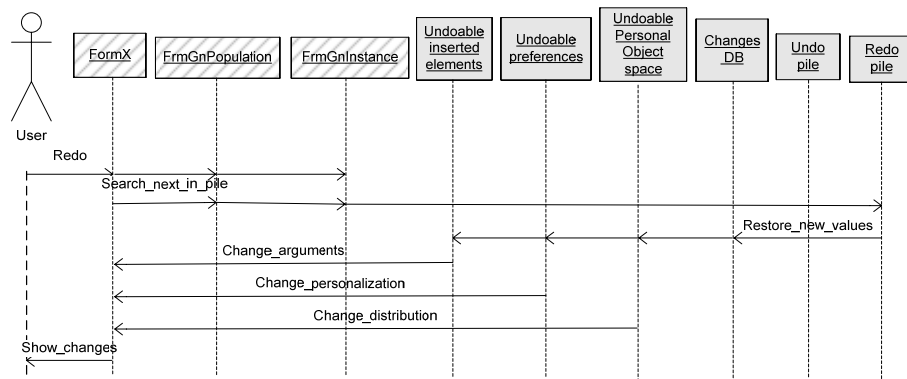


Figura 9.7 Diagrama de Secuencia para incorporar la forma de uso *Rehacer cambios* (rehacer antiguos valores de la pila)

El Diagrama de Secuencia de la Figura 9.7 representa la secuencia de invocación entre métodos cuando el usuario quisiera rehacer algún valor del sistema. Para ello se buscará en la pila cuál es el siguiente valor a rehacer, y se realizarán los cambios oportunos en los campos de entrada, en la

personalización de la interfaz, en la distribución de elementos o en los atributos almacenados en la base de datos.

La Tabla 9.8 muestra un resumen de los cambios que incorpora WU_GU2 al compilador de modelos.

Clase Genérica	Cambios que Incorpora
Form X	Tiene un método para iniciar la acción de rehacer
FrmGnPopulation	Tiene un método para iniciar la acción de rehacer
FrmGnInstance	Tiene un método para iniciar la acción de rehacer
Undoable inserted elements	<ul style="list-style-type: none"> – Almacena los valores previos a ejecutar la acción de deshacer en los campos de entrada – Extrae de la pila los valores previos a la acción deshacer sobre los campos de entrada
Undoable preferences	<ul style="list-style-type: none"> – Almacena el valor de las preferencias previo a la acción de deshacer – Extrae de la pila el valor de las preferencias anteriores a la acción de deshacer
Undoable personal object space	<ul style="list-style-type: none"> – Almacena la distribución de los elementos previa a la ejecución de la acción deshacer – Extrae de la pila la distribución anterior a la ejecución de la acción deshacer
Changes in the data base	<ul style="list-style-type: none"> – Almacena los valores de los atributos de la base de datos antes de que la ejecución de la acción deshacer los modifique – Extrae de la pila los valores de los atributos anteriores a la ejecución de la acción deshacer
Undo pile	Almacena en forma de pila todos los valores previos a la ejecución de la acción deshacer sobre campos de entrada, personalización, distribución de elementos, ejecución de servicios

Clase Genérica	Cambios que Incorpora
MainMenu	Almacena la entrada del menú del sistema mediante la cual acceder a la funcionalidad de rehacer y la combinación de teclas de acceso rápido

Tabla 9.8 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Rehacer cambios*

9.1.4.3 Resumen de los cambios que provoca Global Undo

La Tabla 9.9 muestra un resumen de las formas de uso del mecanismo de usabilidad *Global Undo*, las propiedades que las componen, los modelos conceptuales que se ven afectados, las nuevas primitivas conceptuales necesarias para su representación y los cambios a realizar en el compilador de modelos.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Deshacer cambios	Selección de servicios	Modelo de Objetos	Se deben poder seleccionar los servicios con capacidad de deshacer sus ejecuciones	<ul style="list-style-type: none"> – <i>FormX</i>, <i>FrmGnPopulartion</i>, <i>FrmGnInstance</i> tiene un método para iniciar la acción deshacer – <i>Undoable inserted elements</i> almacena en la pila los valores de los campos de entrada – <i>Changes in the data base</i> almacena en la pila los cambios en la base de datos

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
	Selección de elementos de la interfaz	Ninguno	Ninguno	<ul style="list-style-type: none"> – <i>Undoable preferences</i> almacena las preferencias del usuario en la pila – <i>Undoable personal object space</i> almacena en la pila la distribución de los elementos
	Número máximo de niveles para deshacer	Ninguno	Ninguno	– <i>Undo pile</i> almacena la información en forma de pila
	Forma de acceso	Modelo de Interacción Abstracto	Se debe poder decidir cómo acceder a la funcionalidad de deshacer	<i>MainMenu</i> almacena la forma de acceso a la funcionalidad de deshacer
Rehacer cambios	Selección de servicios	Modelo de Objetos	Se deben poder seleccionar los servicios con capacidad de rehacer las acciones deshechas	<ul style="list-style-type: none"> – <i>Form X</i>, <i>FrmGnPopulartion</i>, <i>FrmGnInstance</i> tiene un método para iniciar la acción de rehacer – <i>Undoable inserted elements</i> almacena los valores deshechos de campos de entrada en la pila

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
				– <i>Changes in the data base</i> almacena en la pila los valores deshechos en los atributos de la base de datos
	Selección de elementos de la interfaz	Ninguno	Ninguno	– <i>Undoable preferences</i> almacena las preferencias deshechas del usuario en la pila – <i>Undoable personal object space</i> almacena en la pila la distribución de los elementos deshecha
	Número máximo de niveles para rehacer	Ninguno	Ninguno	Undo pile almacena la información en forma de pila
	Forma de acceso	Modelo de Interacción Abstracto	Se debe poder decidir cómo acceder a rehacer	<i>MainMenu</i> almacena la forma de acceso a rehacer

Tabla 9.9 Resumen de los cambios para incorporar *Global Undo*

9.2 Abort Operation

Este mecanismo de usabilidad representa la funcionalidad de cancelar una acción antes de que finalice su ejecución. El usuario debe tener la posibilidad de cancelar una acción a mitad ejecución si la acción requiere mucho tiempo (más de 10 segundos). Además, si el usuario entra en un escenario (interfaz) en el cual no desea estar, debería poder salir volviendo al escenario del que procediera (navegación hacia el origen). Esta forma de uso tiene el mismo objetivo que el patrón de usabilidad *Ir un paso atrás* de Tidwell [Tidwell, 2005] y *Salidas de emergencia* de Brighton [Griffiths, 2002].

9.2.1 Formas de uso

A partir de la guía de captura de requisitos del mecanismo de usabilidad *Abort operation* (Anexo I) se obtienen dos formas de uso:

1. WU_AO1: Cancelar durante la ejecución
2. WU_AO2: Salir de un escenario

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Qué acciones pueden necesitar la opción de cancelar?	El usuario debe poder cancelar la ejecución de un servicio que requiera mucho tiempo	Cancelar durante ejecución (WU_AO1)
¿Necesitará el usuario alguna salida en la aplicación?	El usuario debe poder salir de una interfaz para volver a otra anterior	Salir de un escenario (WU_AO2)

Tabla 9.10 Derivación de las formas de uso de *Abort operation* a partir de la guía de captura de requisitos

La Tabla 9.10 contiene las preguntas de la guía de captura de requisitos de la que se han derivado las formas de uso y qué objetivo se pretende conseguir con cada una de ellas.

La primera de las formas de uso, **Cancelar durante la ejecución (WU_AO1)**, se deriva de la pregunta de la guía de captura de requisitos, *¿Qué acciones pueden necesitar la opción de cancelar?* El objetivo de esta forma de uso es el de permitir al usuario cancelar la ejecución de un servicio. Está especialmente pensado para acciones que requieran mucho tiempo. Por ejemplo, en el sistema de alquiler de coches, el usuario debería poder cancelar la reserva de un alquiler mientras se procesa toda la información necesaria para su ejecución.

La otra forma de uso, **Salir de un escenario (WU_AO2)**, se deriva de la pregunta de la guía de captura de requisitos *¿Necesitará el usuario una salida en la aplicación?* El objetivo de esta forma de uso es permitir al usuario volver a un escenario anterior al actual que sea conocido para él. Esta forma de uso está pensada para ayudar al usuario cuando se pierda en la navegación entre escenarios.

Por ejemplo, en el sistema de alquiler de coches, si el usuario entra en la interfaz para dar de alta un cliente desde la interfaz para reservar un alquiler, debe ser capaz de cerrar la primera para volver a la segunda.

En la siguiente sección se explican las propiedades que hay dentro de cada una de las dos formas de uso definidas.

9.2.2 Propiedades

Esta sección explica las propiedades de las dos formas de uso del mecanismo de usabilidad *Abort operation*. Por un lado, las **propiedades de Cancelar durante la ejecución (WU_AO1)** se utilizan para determinar qué servicios se podrán cancelar a mitad ejecución y cómo se presentará dicha funcionalidad.

La Tabla 9.11 presenta las preguntas de la guía de captura de requisitos de las que se han derivado las propiedades y el objetivo que se pretende conseguir con cada una de ellas.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué acciones pueden necesitar la opción de cancelar? ¿Qué acciones requerirán más de diez segundos para su finalización?	Seleccionar los servicios que tendrán la capacidad de cancelar su ejecución	Selección de servicios (P1_WU_AO1)	Configurable
¿Cómo se presentará la opción de cancelar al usuario?	Determinar cómo accederá el usuario a la funcionalidad de cancelar la ejecución del servicio	Especificación del aspecto visual (P2_WU_AO1)	Configurable

Tabla 9.11 Derivación de propiedades de la forma de uso *Cancelar durante la ejecución* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades de WU_AO1 son las siguientes:

- **Selección de servicios (P1_WU_AO1):** Las preguntas de la guía de captura de requisitos de las que se deriva esta propiedad son, *¿Qué acciones pueden necesitar la opción de cancelar? ¿Qué acciones requerirán más de diez segundos para su finalización?* Esta propiedad tiene como objetivo el determinar qué servicios del sistema tendrán la capacidad de cancelar su ejecución. Esta funcionalidad está pensada para servicios cuya ejecución requiera mucho tiempo, ya que si se ejecuta en poco tiempo el usuario no tendría la oportunidad de cancelarla. Es el analista en la fase de análisis el que debe estimar qué servicios serán los más costosos y dotarlos de esta propiedad, por lo tanto, es una propiedad

configurable. Los servicios que tengan asociada la propiedad *Selección de servicios* de la forma de uso *Mostrar progreso de la ejecución*, deberían tener asociado también la propiedad P1_WU_AO1, ya que son propiedades que se complementan. En ambos casos, se deben seleccionar aquellos servicios cuya ejecución requerirá más de diez segundos.

- **Especificación del aspecto visual (P2_WU_AO1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Cómo se presentará la opción de cancelar al usuario?* El objetivo de esta forma de uso es definir cómo se presentará al usuario la opción para cancelar la ejecución de un servicio. Esta propiedad es configurable.

Por ejemplo, en el sistema de alquiler de coches, al servicio reservar alquiler de coche se le podría aplicar la forma de uso *Cancelar durante la ejecución*. De esta manera, mientras se ejecute este servicio, el usuario podría cancelar su ejecución. La propiedad *Selección de servicios* (P1_WU_AO1) tomaría el valor de reservar alquiler de coche y la propiedad *Especificación del aspecto visual* (P2_WU_AO1) tomaría el valor de un botón con la etiqueta *Cancelar*. La Figura 9.8 muestra una posible implementación de estas propiedades.

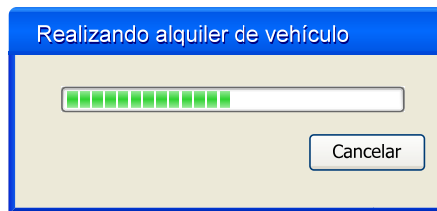


Figura 9.8 Ejecución de reservar alquiler de coche

Por otro lado, **las propiedades de Salir de un escenario (WU_AO2)** determinan las interfaces que permiten regresar a la interfaz desde la que se ha invocado y la manera en que se presenta esa funcionalidad al usuario. La Tabla 9.12 muestra las preguntas de la guía de captura de requisitos de las que se han obtenido las propiedades y los objetivos de cada una de ellas.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Dónde necesitará el usuario una opción de salida?	Determinar las interfaces desde las que el usuario puede retornar a la interfaz que la haya invocado	Selección de las interfaces (P1_WU_AO2)	No configurable
¿Cómo se mostrará al usuario la opción de salida?	Determinar cómo se mostrará la opción de retorno al usuario	Especificación del aspecto visual (P2_WU_AO2)	Configurable

Tabla 9.12 Derivación de propiedades de la forma de uso *Salir de un escenario* a partir de la guía de captura de requisitos

Las propiedades de esta forma de uso son:

- **Selección de las interfaces (P1_WU_AO2):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Dónde necesitará el usuario una opción de salida?* El objetivo de esta propiedad es el especificar qué interfaces tendrán la cualidad de retornar el hilo de ejecución a la interfaz que la haya invocado. Esta propiedad es no configurable porque siguiendo las guías de usabilidad [Tidwell, 2005] [Griffiths, 2002], todas las interfaces deben proporcionar la posibilidad de retornar hacia la ventana de origen de la que proceden. Además, la incorporación de esta propiedad no supone ninguna carga extra para la ejecución del sistema.
- **Especificación del aspecto visual (P2_WU_AO2):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Cómo se mostrará al usuario la opción de salida?* El objetivo de esta propiedad es determinar cómo se visualizará al usuario la funcionalidad de regresar a una interfaz conocida por el usuario.

Esta propiedad es configurable porque el analista debe especificar la manera de visualización más adecuada para cada sistema en particular.

Tomando el sistema de alquiler de coches, se podría incorporar esta forma de uso al servicio de reservar alquiler de coches, de forma que desde la interfaz en la que se crea el cliente se pudiera regresar a la ventana desde la que se realiza el alquiler (alta de cliente se invoca desde reservar alquiler de coche). La Figura 9.9 representa una posible implementación de esta forma de uso. La propiedad *Selección de las interfaces (P1_WU_AO2)* tomaría el valor de la interfaz desde la que se crea al cliente, mientras que la propiedad *Especificación del aspecto visual (P2_WU_AO2)* tomaría el valor de visualizar la opción de salir con forma de botón con una flecha y mediante la combinación de teclas *CTRL+L*.

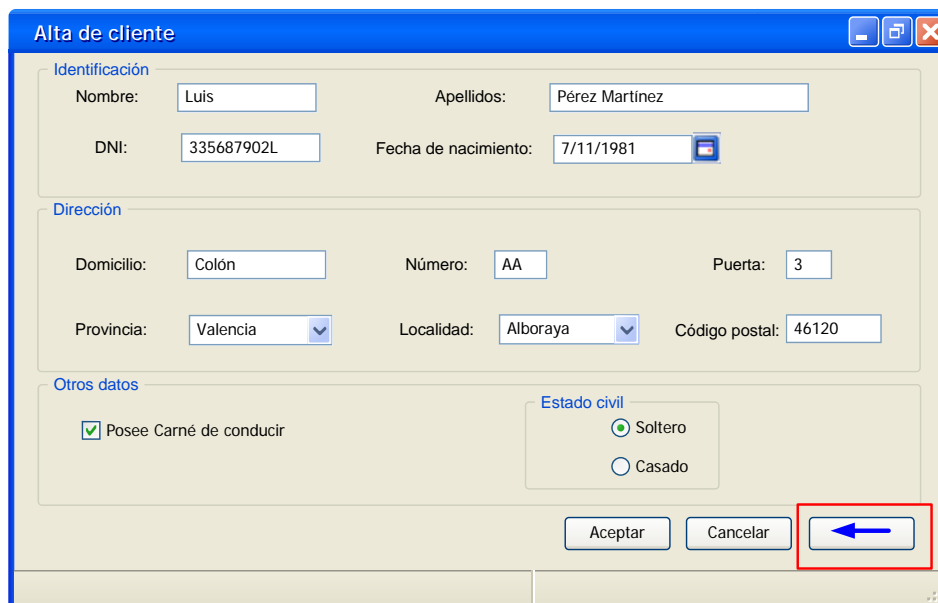


Figura 9.9 Alta de cliente con capacidad para volver a reservar alquiler de coche

9.2.3 Modificación de los modelos conceptuales de OO-Method

Las dos formas de uso del mecanismo de usabilidad *Abort operation* incluyen modificaciones en los modelos conceptuales de OO-Method. Por un lado, la forma de uso **Cancelar durante la ejecución (WU_AO1)** incorpora primitivas conceptuales para seleccionar los servicios que tendrán la posibilidad de cancelar su ejecución y cómo se mostrará esta funcionalidad al usuario. Los cambios que esta forma de uso incorpora son:

- El analista debe poder decidir los servicios que el usuario tendrá la capacidad de cancelar durante su ejecución.
- El analista debe poder decidir la manera en la que la funcionalidad de cancelar el servicio se mostrará al usuario.

En cuanto a la forma de uso **Salir de un escenario (WU_AO2)**, está parcialmente soportada por OO-Method. Actualmente, todas las interfaces poseen un botón *Cancelar* que tiene la funcionalidad de cerrar esa interfaz y devolver el control a la interfaz que la invocó. Esta funcionalidad es la misma que la que propone WU_AO2. Sin embargo, hay que añadir nuevas primitivas conceptuales que permitan modificar la visualización de esta funcionalidad, así el botón *Cancelar* podrá recibir cualquier otra etiqueta (*Atrás*, por ejemplo) o cualquier icono. Por lo tanto, el único cambio es que el analista debe poder decidir el aspecto visual de la funcionalidad para salir de un escenario.

La Tabla 9.13 muestra de forma resumida los cambios que se incorporan a OO-Method a partir de cada una de las propiedades. En la siguiente sección se detallan los cambios a nivel de modelado conceptual que es necesario realizar para incorporar las propiedades configurables a OO-Method.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Cancelar durante la ejecución (WU_AO1)	Selección de servicios (P1_WU_AO1)	Configurable	El analista debe poder decidir los servicios que podrán cancelar su ejecución
	Especificación del aspecto visual (P2_WU_AO1)	Configurable	El analista debe poder decidir cómo visualizar la funcionalidad de cancelar la ejecución del servicio
Salir de un escenario (WU_AO2)	Selección de las interfaces (P1_WU_AO1)	No configurable	No introduce cambios
	Especificación del aspecto visual (P2_WU_AO1)	Configurable	El analista debería poder decidir cómo visualizar la salida de las interfaces

Tabla 9.13 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Abort operation*

9.2.3.1 WU_AO1: Cancelar durante la ejecución

La forma de uso *Cancelar durante la ejecución (WU_AO1)* introduce cambios en dos modelos de OO-Method. Por un lado, la propiedad *Selección de servicios (P1_WU_AO1)* afecta al **Modelo de Objetos**. En este modelo el analista debe especificar los servicios que tendrán la capacidad de cancelar su ejecución. Para ello se debe incorporar una primitiva conceptual para cada servicio que indique si tiene o no la capacidad de cancelarse durante su ejecución. Es lógico que esta primitiva

conceptual se añade al Modelo de Objetos porque tiene implicaciones funcionales, y es en el momento de definición de los servicios donde el analista puede intuir qué servicios requerirán más tiempo para su ejecución y, por tanto, deban tener la capacidad de cancelar su ejecución. Las transacciones son los elementos más propicios para incorporar la capacidad de cancelar porque pueden contener un gran número de servicios o incluso otras transacciones. Todo servicio al que se le habilite la posibilidad de cancelación debe llevar una barra de progreso, ya que si se habilita la cancelación significa que su ejecución llevará bastante tiempo. Por lo tanto, aun sin el consentimiento del analista, todo servicio que incorpore la funcionalidad de la forma de uso *Cancelar durante la ejecución*, incorporará también la funcionalidad de *Mostrar progreso de la ejecución*.

Por otro lado, la propiedad *Especificación del aspecto visual (P2_WU_AO1)* implica añadir nuevas primitivas al **Modelo de Interacción Concreto** para definir cómo se visualizará la posibilidad de cancelar una acción durante su ejecución. Las primitivas conceptuales a añadir son:

- La ubicación del botón
 - Se muestra en la ventana emergente (junto a la barra de progreso)
 - Se muestra en la ventana principal
 - No se muestra
- El aspecto visual del botón
 - Icono. En este caso el analista debe especificar el icono
 - Texto. En este caso el analista debe especificar el texto y su formato
 - Icono junto a texto. En este caso el analista debe especificar tanto el icono como el texto y su formato
- Las teclas de acceso rápido a través de las cuales se puede llevar a cabo la cancelación

Por defecto, ningún servicio se podrá cancelar durante su ejecución. La visualización por defecto será mostrar el botón de cancelación en la ventana emergente que muestra la barra de progreso mediante texto en fuente Arial, tamaño 10, color negro y alineación centrada. Las teclas de acceso rápido son *CTRL+A*.

La Tabla 9.14 muestra un resumen de los cambios que incorpora en OO-Method la forma de uso *Cancelar durante la ejecución*.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Objetos	Selección de los servicios con la capacidad de cancelar su ejecución
Modelo de Interacción Concreto	<ul style="list-style-type: none">– Dónde se ubicará el botón de cancelación– Aspecto visual del botón (icono y texto con formato)– Combinación de teclas para acceso rápido

Tabla 9.14 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Cancelar durante la ejecución*

9.2.3.2 WU_AO2: Salir de un escenario

La forma de uso *Salir de un escenario (WU_AO2)* sólo incorpora primitivas conceptuales para modelar cómo se representará al usuario la salida de las interfaces. Por lo tanto, sólo afectaría al **Modelo de Interacción Concreto**. Este modelo incorpora primitivas conceptuales para determinar cada uno de los siguientes aspectos de la forma de uso:

- La posición del botón de salida en la interfaz
- El texto del botón de salida
 - Contenido del texto
 - Formato del texto
- El icono del botón de salida
- Combinación de teclas de acceso rápido a la funcionalidad de salida

Por defecto, se mostrará un botón en el margen inferior derecho de la interfaz, con el texto *Salir*, y se podrá acceder a su funcionalidad con la combinación de teclas *CTRL+S*.

La Tabla 9.15 muestra de forma resumida los cambios que incorpora *Salir de un escenario* a los modelos conceptuales de OO-Method.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Concreto	<ul style="list-style-type: none"> – Posición del botón de salida – Texto del botón de salida – Icono del botón de salida – Combinación de teclas para acceso rápido

Tabla 9.15 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Salir de un escenario*

A continuación se presentan los cambios que habría que introducir en el compilador de modelos para que el código generado reflejara la configuración que el analista haya definido mediante estas primitivas.

9.2.4 Modificación del compilador de modelos

9.2.4.1 WU_AO1: Cancelar durante la ejecución

La Figura 9.10 presenta el Diagrama de Clases donde se representa cómo introducir las formas de uso *Cancelar durante la ejecución (WU_AO1)* y *Salir de un escenario (WU_AO2)*. A continuación se explican los cambios que provoca WU_AO1 y en la siguiente sección se contarán los cambios que provoca WU_AO2.

La forma de uso *Cancelar durante la ejecución* se incorpora al sistema mediante la clase *Cancel execution button* que representa el botón a través del cual se cancela la ejecución. Este botón se puede representar bien en el formulario donde se presente la barra de progreso (clase *Form_ProgressBar*) o bien en la ventana principal (clase *FrmMDIMain*). También puede que este botón estuviera oculto para el usuario y que sólo se pudiera ejecutar mediante teclas de acceso rápido. Las clases que

detienen la ejecución una vez el usuario haya invocado esta petición son *Service wrapper* y *ClassX action*.

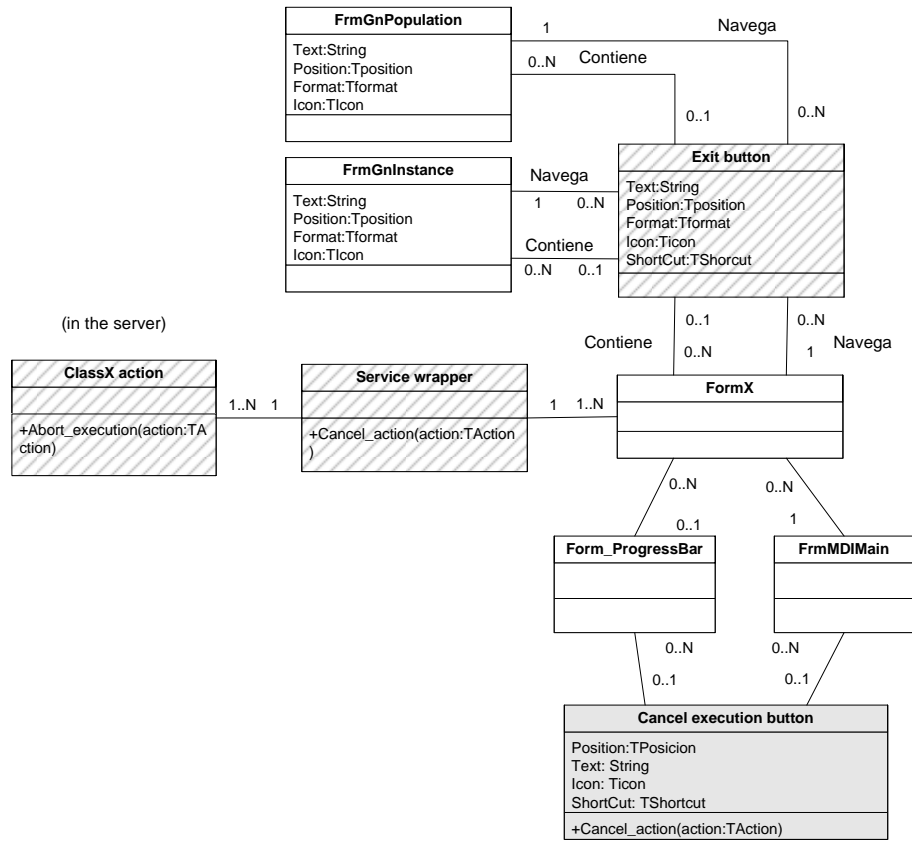


Figura 9.10 Diagrama de Clases para *Abort Operation*

La Figura 9.11 muestra el Diagrama de Secuencia donde se puede visualizar la secuencia de llamadas entre las clases mostradas en el Diagrama de Clases anterior. Este diagrama representa la operación de cancelación visualizada en el mismo formulario donde se muestra la barra de progreso (clase *Form_ProgressBar*). El usuario iniciará la cancelación mediante el servicio *Cancel_action* de la clase *Cancel execution button*. La clase que cancelará la ejecución del servicio es *ClassX action*, ya que es la clase que implementa su lógica de negocio.

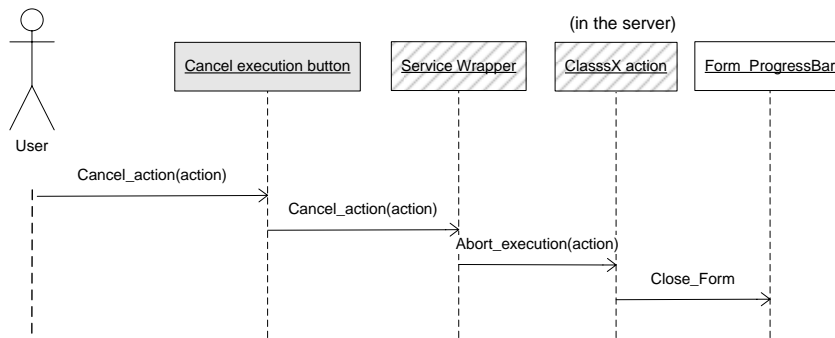


Figura 9.11 Diagrama de Secuencia para incorporar la forma de uso *Cancelar durante la ejecución*

La Tabla 9.16 muestra un resumen de los cambios que incorpora *Cancelar durante la ejecución* al compilador de modelos de OO-Method.

Clase Genérica	Cambios que Incorpora
Cancel execution button	<ul style="list-style-type: none"> – Inicia la invocación para cancelar la ejecución del servicio – Almacena el aspecto visual de la forma de acceso a la funcionalidad de cancelar
Service wrapper	Recibe la petición de cancelación de la clase <i>Cancel execution button</i> y la transmitiría al servidor
Class X action	Ejecuta la cancelación del servicio

Tabla 9.16 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Cancelar durante la ejecución*

9.2.4.2 WU_AO2: Salir de un escenario

El Diagrama de Clases de la Figura 9.10 incluye las modificaciones que hay que incorporar al compilador de modelos para dar soporte a la forma de uso *Salir de un escenario*. La clase *Exit button* ya existe actualmente en el botón de *Cancelar*. La modificación de esta clase consiste en añadirle atributos para que el analista pueda decidir sobre el aspecto visual de la funcionalidad. De esta manera la clase almacena: el texto del botón, su formato, la posición, el icono y la combinación de teclas rápidas para

acceder a su funcionalidad. El botón de salida se puede encontrar en interfaces derivadas de una UIS (clase *Form X*), una UIP (clase *FrmGnPopulation*), o una UII (clase *FrmGnInstance*).

Esta forma de uso no dispone de Diagrama de Secuencia porque su funcionalidad no implica realizar nuevas invocaciones entre los métodos de las clases, sólo incorpora nuevos atributos y relaciones a la clase *Exit button*.

La Tabla 9.17 muestra el único cambio que se debe incorporar al compilador de modelos de OO-Method para dar soporte a *Salir de un escenario*

Clase Genérica	Cambios que Incorpora
Exit button	Almacena los aspectos de visualización del botón de salida de las interfaces

Tabla 9.17 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Salir de un escenario*

9.2.4.3 Resumen de los cambios que provoca Abort Operation

La Tabla 9.18 muestra un resumen de las formas de uso del mecanismo de usabilidad *Global Undo*, las propiedades que las componen, los modelos conceptuales que se ven afectados, las nuevas primitivas conceptuales necesarias para su representación y los cambios a realizar en el compilador de modelos para implementar el mecanismo de usabilidad.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Cancelar durante la ejecución	Selección de servicios	Modelo de Objetos	Selección de servicios con capacidad de cancelar su ejecución	– <i>Cancel execution button</i> inicia la cancelación

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
				<ul style="list-style-type: none"> – <i>Service wrapper</i> transmite la petición al servidor – <i>Class X action</i> ejecuta la cancelación
	Especificación del aspecto visual	Modelo de Interacción Concreto	Definir dónde se ubicará el botón, su aspecto y las teclas de acceso rápido	<i>Cancel execution button</i> almacena el aspecto visual del botón
Salir de un escenario	Selección de las interfaces	Ninguno	Ya soportada	Ninguno
	Especificación del aspecto visual	Modelo de Interacción Concreto	Posición del botón de salida, su texto, su icono y las teclas de acceso rápido	<i>Exit button</i> almacena los aspectos de visualización del botón de salida

Tabla 9.18 Resumen de los cambios para incorporar *Abort Operation*

Capítulo 10

Incorporación de Help

Help es un FUF que tiene como objetivo proporcionar ayuda al usuario sobre cómo llevar a cabo las acciones del sistema. *Help* tiene un único mecanismo de usabilidad llamado *Multilevel help*. Este capítulo tiene una única sección en la que se explica el resultado de aplicar el método MIMAT a dicho mecanismo de usabilidad.

10.1 Multilevel Help

Este mecanismo de usabilidad introduce la funcionalidad para ayudar al usuario sobre el funcionamiento del sistema. Es útil para aquellos sistemas que sean complejos y que vayan a ser utilizados por un gran número de usuarios con distintos niveles en el conocimiento del dominio de la aplicación. La ayuda está pensada para los usuarios novatos y rara vez es utilizada por los expertos. Por lo tanto, la visibilidad de la ayuda debe ser una característica que defina el usuario dependiendo de sus conocimientos. Otra característica que debe tener la ayuda es que se debe redactar de forma breve y clara, sino el usuario no la leerá.

10.1.1 Formas de uso

De la guía para la captura de requisitos del mecanismo de usabilidad *Multilevel help* (Anexo I) se obtienen las siguientes formas de uso:

1. WU_MH1: Ayuda dinámica
2. WU_MH2: Ayuda estática

La Tabla 10.1 presenta una visión global de las formas de uso, las preguntas de la guía de captura de requisitos de las que se han derivado y los objetivos que se pretenden conseguir con ellas.

Pregunta de la guía	Objetivo de la forma de uso	Forma de uso
¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?	Mostrar ayuda dependiendo de la acción que realice el usuario	Ayuda dinámica (WU_MH1)
¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?	Mostrar ayuda detallada de toda la funcionalidad del sistema	Ayuda estática (WU_MH2)

Tabla 10.1 Derivación de las formas de uso de *Multilevel help* a partir de la guía de captura de requisitos

La primera de las formas de uso, **Ayuda dinámica (WU_MH1)**, se deriva de la pregunta de la guía de captura de requisitos, *¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?* El objetivo de esta forma de uso es determinar la ayuda que emergerá automáticamente mientras el usuario esté interactuando con el sistema. Esta ayuda sólo muestra una explicación del escenario en el cual se encuentra el usuario en ese momento concreto. Un ejemplo de este tipo de ayuda son los *Tooltips* que se muestran cuando el usuario pasa el puntero del ratón por encima de algún elemento de la interfaz. La ayuda que se muestra de esta manera requiere que el texto descriptivo sea muy breve. Otro ejemplo de este tipo de ayuda es la que aparece cuando el usuario selecciona algún elemento de la interfaz y aparece automáticamente el contenido de la ayuda en un lado de la interfaz. En este caso, el texto descriptivo puede ser más detallado que en el caso de los *Tooltip*.

Por ejemplo, en el sistema de alquiler de coches, el servicio poner en venta se lanzaría desde un botón. Cuando el usuario pasara el ratón sobre el

botón se podría mostrar un texto explicativo de la funcionalidad de ese botón.

La segunda forma de uso, **Ayuda estática (WU_MH2)**, se deriva también de la pregunta de la guía de captura de requisitos *¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?* El objetivo de esta forma de uso es el de mostrar ayuda del sistema de forma detallada y no de forma breve como WU_MH1. Esta ayuda es la que contiene información detallada de toda la aplicación y normalmente se accede a ella a partir de una entrada desde el menú principal de la aplicación. El contenido de la ayuda se mostraría en una interfaz diferente de las interfaces de la aplicación y suele estar escrita en HTML, aunque a veces también aparece en forma de WinHelp. En este tipo de ayuda también entrarían los manuales en línea que se acceden vía Internet.

Por ejemplo, en el sistema de alquiler de coches se podría añadir una entrada del menú principal llamada *Ayuda* que detallara toda la funcionalidad del sistema.

En la siguiente sección se van a explicar las propiedades que componen cada una de estas formas de uso.

10.1.2 Propiedades

En esta sección se detallan las propiedades en las que se dividen las dos formas de uso de *Multilevel help*. **Las propiedades de Ayuda dinámica (WU_MH1)** especifican los elementos de la interfaz que dispondrán de ayuda dinámica y cómo se presentará esa ayuda al usuario.

La Tabla 10.2 muestra de forma resumida cómo se han derivado las propiedades a partir de las preguntas de la guía de captura de requisitos.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Los usuarios que usen el sistema serán expertos o novatos?	La ayuda dinámica se debería poder habilitar dependiendo del grado de conocimiento del usuario	Habilitar ayuda dinámica (P1_WU_MH1)	Configurable
¿Qué clase de tareas requerirán ayuda para realizarse?	Se definen los elementos de la interfaz que tendrán ayuda dinámica	Elementos de la interfaz (P2_WU_MH1)	Configurable
¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?	Se define el texto que se mostrará en cada elemento de la interfaz con ayuda dinámica	Contenido de la ayuda (P3_WU_MH1)	Configurable
¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?	Determinar cómo se mostrará la ayuda dinámica al usuario	Visualización de la ayuda (P4_WU_MH1)	Configurable

Tabla 10.2 Derivación de propiedades de la forma de uso *Ayuda dinámica* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades son las siguientes:

- **Habilitar ayuda dinámica (P1_WU_MH1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Los usuarios que usen el sistema serán expertos o novatos?* El objetivo de esta propiedad es habilitar o no la ayuda que se muestra dinámicamente mientras el usuario interactúa con el sistema. Esta ayuda dinámica debe estar habilitada si el usuario que interactúa con el sistema es una persona novata, ya que le puede ser de gran utilidad. En cambio, los usuarios expertos no la utilizan y puede que incluso llegue a entorpecer su labor porque incrementa la cantidad de información de la interfaz. Además, la repetición de las explicaciones puede llegar a causar irritación en el usuario. Esta propiedad es configurable, es decir, el analista habilita o no la ayuda dinámica. También se podría dejar esta decisión en mano del usuario, de manera que habilitara la ayuda dinámica cuando la necesitara.
- **Elementos de la interfaz (P2_WU_MH1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué clase de tareas requerirán ayuda para realizarse?* El objetivo de esta propiedad es determinar los elementos de la interfaz que llevarán asociada la ayuda dinámica. Esta propiedad es configurable.
- **Contenido de la ayuda (P3_WU_MH1):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?* El objetivo de esta propiedad es definir el contenido de la ayuda asociado a cada elemento de la interfaz. Esta propiedad es configurable, el contenido lo especifica el analista.
- **Visualización de la ayuda (P4_WU_MH1):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?* El objetivo de esta propiedad es el de especificar cómo se mostrará la ayuda dinámica al usuario. La ayuda se puede mostrar de distinta forma. Por ejemplo, mediante *Tooltips* o con texto que se muestre en alguna parte de la interfaz. La ayuda que se visualiza

con *Tooltips* está pensada para ayudas con poco contenido de texto, mientras que la ayuda con texto emergente puede contener más cantidad de caracteres. Esta propiedad es configurable ya que el tipo de visualización lo elige el analista.

Como ejemplo para ilustrar el uso de estas propiedades se ha elegido el sistema de alquiler de coches. Si definiéramos una ayuda dinámica para el servicio de poner en venta, se mostraría un mensaje de ayuda cada vez que el usuario pasara el puntero del ratón sobre el botón que lanzase dicho servicio. La Figura 10.1 muestra una posible implementación de la forma de uso *Ayuda dinámica*. Para implementar este ejemplo, la propiedad *Habilitar ayuda dinámica (P1_WU_MH1)* tendría el valor de habilitada; *Elementos de la interfaz (P2_WU_MH1)* tendría el valor del botón que ejecutaría el servicio poner en venta; *Contenido de la ayuda (P3_WU_MH1)* tendría el valor “El coche deja de poderse alquilar y pasa a ventas”; y *Visualización de la ayuda (P4_WU_MH1)* tendría el valor de mostrar el texto mediante *Tooltip*.

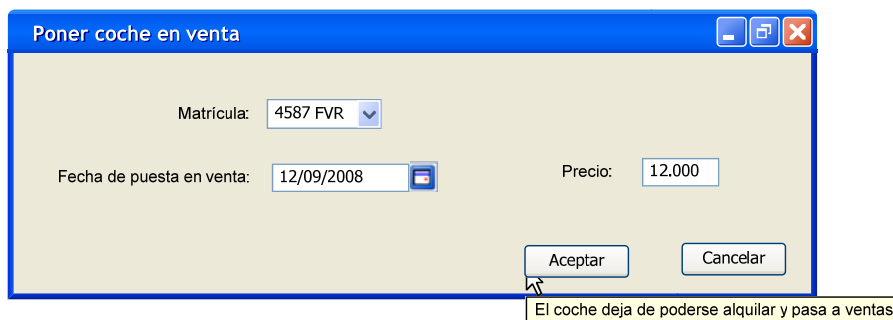


Figura 10.1 Poner en venta con ayuda dinámica

Por otro lado, **las propiedades de Ayuda estática (WU_MH2)** definen el contenido de la ayuda que explicará el funcionamiento de todo el sistema y cómo se accederá a ella. La Tabla 10.3 presenta las preguntas de la guía de captura de requisitos de la que se han derivado las propiedades y los objetivos que se pretenden conseguir con ellas. Como se puede apreciar, todas las propiedades se han derivado de la misma pregunta, pero cada una de ellas tiene un objetivo distinto.

Pregunta de la guía	Objetivo de la propiedad	Propiedad	Tipo
¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?	Especificar el texto que se mostrará en la ayuda	Contenido de la ayuda (P1_WU_MH2)	Configurable
¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?	Definir cómo accederá el usuario a la ayuda	Acceso a la ayuda (P2_WU_MH2)	Configurable
¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?	Definir aspectos de formato del texto de ayuda	Visualización de la ayuda (P3_WU_MH2)	Configurable

Tabla 10.3 Derivación de propiedades de la forma de uso *Ayuda estática* a partir de la guía de captura de requisitos

Más detalladamente, las propiedades de WU_MH2 son:

- **Contenido de la ayuda (P1_WU_MH2):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?* El objetivo de esta propiedad es el de definir la información que se mostrará al usuario cuando éste solicite la ayuda del sistema. La ayuda debe abarcar toda la funcionalidad del sistema y debe tener en cuenta tanto a los usuarios noveles como a los expertos. Es una propiedad configurable.

- **Acceso a la ayuda (P2_WU_MH2):** La pregunta de la guía de captura de requisitos de la que se deriva esta propiedad es, *¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?* Esta propiedad determina en qué forma el usuario podrá acceder a la ayuda. Es una propiedad configurable porque es el analista el que debe especificar la forma de acceder a la ayuda (mediante qué interfaces, mediante qué entrada del menú, mediante qué combinación de teclas, etc.)
- **Visualización de la ayuda (P3_WU_MH2):** Esta propiedad se deriva de la pregunta de la guía de captura de requisitos, *¿Qué clase de ayuda es la más adecuada para cada una de las tareas del sistema?* Esta pregunta es la misma que la utilizada para extraer la propiedad *Acceso a la ayuda*, pero su objetivo es distinto. En este caso, la propiedad tiene el objetivo de determinar cómo se visualizará el contenido de la ayuda al usuario, es decir, aspectos de formato del texto de la ayuda.

El sistema de alquiler de coches podría contener una sección de ayuda donde se explicara toda su funcionalidad. Las propiedades tomarían los siguientes valores para este ejemplo: *Contenido de la ayuda (P1_WU_MH2)* tendría un texto descriptivo de toda la funcionalidad del sistema agrupado por interfaces (por ejemplo); *Acceso a la ayuda (P2_WU_MH2)* definiría el acceso por medio de una entrada *Ayuda* en el menú principal y mediante la tecla de acceso rápido *F1*; *Visualización de la ayuda (P3_WU_MH2)* se habría definido con un formato Arial, alineación justificada y tamaño de letra 12.

10.1.3 Modificación de los modelos conceptuales de OO-Method

Las dos formas de uso tienen propiedades configurables y, por lo tanto, ambas introducen cambios a nivel de modelado conceptual porque no están soportadas actualmente por OO-Method de forma completa. Por lo que respecta a la forma de uso **Ayuda dinámica (WU_MH1)**, sólo está hoy en día disponible en los Patrones Elementales de Acción. Cada vez que el usuario pasa el puntero del ratón por los botones de una UIP o una UII, se

muestra mediante un *Tooltip* el nombre del servicio que se ejecuta al pulsar dicho botón. Esto ayuda a diferenciar los distintos botones entre sí, ya que la mayoría de las veces los iconos de estos botones coinciden incluso dentro de la misma interfaz. Se debe que hacer extensible esta forma de uso al resto de elementos de la interfaz (no sólo sobre los Patrones Elementales de Acción).

La propiedad *Contenido de la ayuda* está soportada por todos los Patrones Elementales del Modelo de Interacción Abstracto. Hoy en día ya existen primitivas para que el analista introduzca el contenido de la ayuda a nivel de modelo, pero dicho texto no se muestra al usuario. El contenido de estos mensajes sólo sirve actualmente para guiar la labor del analista. Por lo tanto, no hay que añadir nuevas primitivas para representar la propiedad *Contenido de la ayuda*, pero sí hay que modificar el compilador de modelos para que el contenido de estos mensajes de ayuda se muestre al usuario.

Por tanto, los cambios a incorporar en el modelado conceptual son:

- El analista o el usuario final deben poder habilitar o inhabilitar la ayuda dinámica, dependiendo de si el sistema es utilizado por usuarios expertos o noveles
- El analista debe poder decidir los elementos de la interfaz que llevarán asociada la ayuda dinámica
- El compilador de modelos debe incluir en las aplicaciones generadas el texto de ayuda representado en primitivas
- El analista debe poder decidir cómo visualizar la ayuda al usuario (con *Tooltips* o mediante texto que se mostraría en alguna sección de la interfaz).

En cuanto a la otra forma de uso, **Ayuda estática (WU_MH2)**, no está soportada en ningún porcentaje por OO-Method. Los sistemas generados no incorporan ningún manual o ayuda donde se explique su funcionalidad. Los cambios que hay que realizar en el modelado conceptual para darle soporte son:

- El analista debe poder introducir el contenido de la ayuda

- El analista debe poder especificar cómo podrá acceder el usuario a la ayuda (desde el menú del sistema, desde las interfaces y con teclas de acceso rápido)
- El analista debe poder especificar el formato con el se visualizará la ayuda al usuario

La Tabla 10.4 muestra de forma resumida los cambios que hay que incorporar a OO-Method para soportar las propiedades de las formas de uso de *Multilevel help*.

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Ayuda dinámica (WU_MH1)	Habilitar ayuda dinámica (P1_WU_MH1)	Configurable	El analista y el usuario deben poder habilitar o inhabilitar la ayuda dinámica cuando lo consideren oportuno
	Elementos de la interfaz (P2_WU_MH1)	Configurable	El analista debe poder determinar los elementos de la interfaz que tendrán ayuda dinámica
	Contenido de la ayuda (P3_WU_MH1)	Configurable	El compilador de modelos debe incluir el texto de ayuda en las aplicaciones generadas
	Visualización de la ayuda (P4_WU_MH1)	Configurable	El analista debe poder decidir cómo visualizar la ayuda dinámica

Forma de uso	Propiedad	Tipo	Cambios en OO-Method
Ayuda estática (WU_MH2)	Contenido de la ayuda (P1_WU_MH2)	Configurable	El analista debe poder introducir el contenido de la ayuda estática
	Acceso a la ayuda (P2_WU_MH2)	Configurable	El analista debe poder definir cómo accederá el usuario a la ayuda
	Visualización de la ayuda (P3_WU_MH2)	Configurable	El analista debe poder seleccionar el formato con el que se presentará la ayuda

Tabla 10.4 Resumen de los cambios en OO-Method que produce cada una de las propiedades de *Multilevel help*

En la siguiente sección se detallan los cambios que se deben realizar a nivel de modelado y del compilador de modelos para que OO-Method pueda dar pleno soporte a las dos formas de uso

10.1.3.1 WU_MH1: Ayuda dinámica

Las propiedades configurables de *Ayuda dinámica (WU_MH1)* que introducen cambios en el modelado conceptual son: *Habilitar ayuda dinámica (P1_WU_MH1)*; *Elementos de la interfaz (P2_WU_MH1)* y *Visualización de la ayuda (P4_WU_MH1)*. Por un lado, las propiedades *Habilitar ayuda dinámica (P1_WU_MH1)* y *Elementos de la interfaz (P2_WU_MH1)* afectan al **Modelo de Interacción Abstracto**, ya que especifican qué ayuda se va a mostrar al usuario. Son necesarias nuevas primitivas conceptuales para representar las siguientes características:

- *Habilitación de la ayuda dinámica en todo el sistema (el analista y el usuario deberían ser capaces de habilitar o no la ayuda dinámica).*

- Selección de elementos de la interfaz con ayuda dinámica. Los posibles elementos serían:
 - Argumentos de entrada
 - Variables de filtro
 - Criterios de ordenación
 - Atributos del conjunto de visualización
 - Acciones
 - Navegaciones

Por defecto, ningún elemento de la interfaz llevará asociada ayuda dinámica y la ayuda estará inhabilitada.

En cuanto a la propiedad restante, *Visualización de la ayuda (P4_WU_MH1)*, su incorporación sólo afecta al **Modelo de Interacción Concreto** porque esta propiedad representa cómo se mostrará la ayuda dinámica al usuario. Se debe añadir una única primitiva que represente la manera de mostrar la ayuda. Las posibles alternativas son:

- Mostrar ayuda mediante *Tooltips*, es decir, texto emergente en una barra amarilla cuando el usuario pase el puntero del ratón sobre el elemento con ayuda definida.
- Mostrar ayuda mediante la incorporación de texto en la propia interfaz. En este caso la ayuda se mostrará también cada vez que el usuario ponga el puntero del ratón encima del elemento de la interfaz con ayuda dinámica. Se diferencia del anterior tipo de visualización en que en este caso el contenido no se muestra en una barra amarilla sobre el elemento, sino en alguna porción de la interfaz destinada a la visualización del texto de ayuda. Cada vez que el usuario pase el ratón por encima de un elemento con ayuda, en la porción elegida se mostrará el texto asociado a esa ayuda. Para visualizar la ayuda dinámica, habrá una región de la interfaz que compartirán todos los elementos con ayuda definida. Para modelar esta forma de visualización es necesaria otra primitiva conceptual para representar en qué posición de la interfaz se mostrará el contenido de la ayuda. Este tipo de ayuda permite un mayor número de caracteres que la ayuda mediante *Tooltips*.

Por defecto, la ayuda se mostrará mediante *Tooltips* porque interfiere menos en el trabajo del usuario que si se muestra gran cantidad de texto en la interfaz. La Tabla 10.5 muestra de forma resumida los cambios que incorpora *Ayuda dinámica* a los modelos conceptuales de OO-Method.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Abstracto	<ul style="list-style-type: none"> – Habilitar o no la ayuda dinámica – Seleccionar elementos con ayuda dinámica
Modelo de Interacción Concreto	Visualización con <i>Tooltips</i> o con texto en alguna porción de la interfaz (en este último caso se debe añadir otra primitiva para especificar la posición del contenido de la ayuda)

Tabla 10.5 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso *Ayuda dinámica*

10.1.3.2 WU_MH2: Ayuda estática

Esta forma de uso también implica cambios en el modelado conceptual de OO-Method, aunque son cambios menores. Las propiedades *Contenido de la ayuda (P1_WU_MH2)* y *Visualización de la ayuda (P3_WU_MH2)* no se van a incorporar a nivel conceptual. Normalmente, la ayuda estática en los sistemas consiste en varias páginas HTML que contienen información sobre el funcionamiento de la aplicación. Por tanto, el contenido de la ayuda y sus opciones de visualización se pueden definir en cualquier herramienta de las que ya existen hoy en día para tal fin y es algo totalmente independiente de OO-Method.

Por otro lado, la propiedad *Acceso a la ayuda (P2_WU_MH2)* sí que introduce cambios en el modelo conceptual de OO-Method. Hay que modificar el **Modelo de Interacción Abstracto** para que el analista pueda añadir accesos a la ayuda estática. Este acceso se debe poder realizar desde el menú principal o desde teclas de acceso rápido. Por tanto, el Modelo de Interacción Abstracto debe añadir una nueva primitiva para

representar la ayuda en el menú principal y otra para representar su acceso mediante una combinación de teclas.

Por defecto, la ayuda se accederá desde la entrada del menú llamada *Ayuda* y mediante la tecla *F1*. La Tabla 10.6 muestra un resumen de los cambios que incorpora *Ayuda estática* a los modelos conceptuales de OO-Method.

Modelo de OO-Method	Nueva Primitiva Conceptual
Modelo de Interacción Abstracto	Acceso a la ayuda estática

Tabla 10.6 Resumen de la nueva primitivas conceptual para incorporar la forma de uso *Ayuda estática*

A continuación se presentan los cambios a introducir en el compilador de modelos para que el código generado refleje la configuración que el analista haya definido mediante estas primitivas.

10.1.4 Modificación del compilador de modelos

10.1.4.1 WU_MH1: Ayuda dinámica

Es necesario modificar el compilador de modelos para que los mensajes de ayuda asociados a primitivas del Modelo de Interacción Abstracto de OO-Method (que representan elementos de la interfaz) se muestren dinámicamente al usuario.

La Figura 10.2 muestra el Diagrama de Clases con las clases que se ven afectadas para soportar las primitivas donde se introduce el contenido de la ayuda. Los cambios a realizar son: añadir un atributo llamado *Available* que indique si la ayuda está o no habilitada; un atributo *Help* que almacene el contenido de la ayuda dinámica; un atributo llamado *Visualization* que almacene los opciones de visualización de la ayuda dinámica. Para esta forma de uso no es necesario utilizar Diagramas de Secuencia para ilustrar

los cambios en el compilador de modelos porque no afecta a ningún servicio.

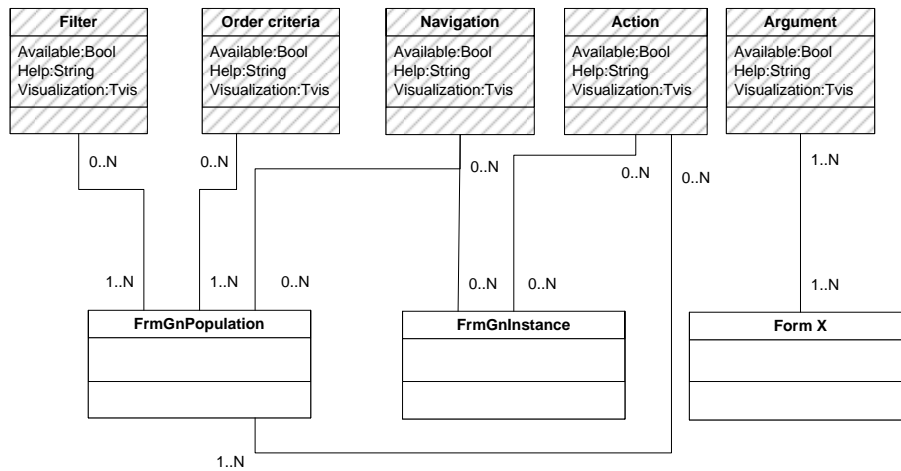


Figura 10.2 Diagrama de Clases para *Multilevel Help*

La Tabla 10.7 muestra un resumen de los cambios que incorpora *Ayuda dinámica* al compilador de modelos de OO-Method.

Clase Genérica	Cambios que Incorpora
Filter	<ul style="list-style-type: none"> – Indicar si la ayuda dinámica está habilitada o no – Almacenar el texto que se mostrará en la ayuda – Almacenar las opciones de visualización de la ayuda
Order criteria	<ul style="list-style-type: none"> – Indicar si la ayuda dinámica está habilitada o no – Almacenar el texto que se mostrará en la ayuda – Almacenar las opciones de visualización de la ayuda
Navigation	<ul style="list-style-type: none"> – Indicar si la ayuda dinámica está habilitada o no – Almacenar el texto que se mostrará en la ayuda – Almacenar las opciones de visualización de la ayuda

Clase Genérica	Cambios que Incorpora
Action	<ul style="list-style-type: none"> – Indicar si la ayuda dinámica está habilitada o no – Almacenar el texto que se mostrará en la ayuda – Almacenar las opciones de visualización de la ayuda
Argument	<ul style="list-style-type: none"> – Indicar si la ayuda dinámica está habilitada o no – Almacenar el texto que se mostrará en la ayuda – Almacenar las opciones de visualización de la ayuda

Tabla 10.7 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso *Ayuda dinámica*

10.1.4.2 WU_MH2: Ayuda estática

Las propiedades de la forma de uso *Ayuda estática (WU_MH2)* también incorporan cambios en el compilador de modelos. El único cambio afecta a la clase *MainMenu*, donde se debe almacenar la forma en la que se accederá a la ayuda estática tanto si se accede desde el menú principal como si se accede mediante teclas de acceso rápido. El resto de cambios en el compilador de modelos no afectan a la arquitectura del sistema generado, ya que la funcionalidad de esta forma de uso se implementa con una serie de ficheros en HTML que contienen el texto de la ayuda estática en el formato deseado por el analista. Estos ficheros se abrirán cada vez que el usuario solicite la ayuda. Al no afectar de manera significativa a la arquitectura del sistema, los cambios que provoca la incorporación de *Ayuda estática* no se ha representado con Diagramas de Clase ni de Secuencia.

10.1.4.3 Resumen de los cambios que provoca Multilevel Help

La Tabla 10.8 muestra un resumen de las formas de uso del mecanismo de usabilidad *Multilevel Help*, las propiedades que las componen, los modelos conceptuales que se ven afectados, las nuevas primitivas conceptuales

necesarias para su representación y los cambios a realizar en el compilador de modelos para implementar el mecanismo de usabilidad.

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
Ayuda dinámica	Habilitar ayuda dinámica	Modelo de Interacción Abstracto	Habilitar o no la ayuda dinámica	Las clases <i>Filter</i> , <i>Order criteria</i> , <i>Navigation</i> , <i>Action</i> , <i>Argument</i> tienen un atributo que representa si la ayuda está habilitada o no
	Elementos de la interfaz	Modelo de Interacción Abstracto	Seleccionar elementos de la interfaz con ayuda dinámica	Las clases <i>Filter</i> , <i>Order criteria</i> , <i>Navigation</i> , <i>Action</i> , <i>Argument</i> indican los elementos con ayuda
	Contenido de la ayuda	Modelo de Interacción Abstracto	Contenido de la ayuda dinámica	Las clases <i>Filter</i> , <i>Order criteria</i> , <i>Navigation</i> , <i>Action</i> , <i>Argument</i> tienen un atributo que representa el contenido de la ayuda

Formas de uso	Propiedades	Modelo afectado	Cambios conceptuales	Cambios compilador
	Visualización de la ayuda	Modelo de Interacción Concreto	Cómo se visualizará la ayuda (<i>Tooltips</i> o texto en la interfaz)	Las clases <i>Filter</i> , <i>Order criteria</i> , <i>Navigation</i> , <i>Action</i> , <i>Argument</i> tienen un atributo para representar las opciones de visualización
Ayuda estática	Acceso a la ayuda	Modelo de Interacción Concreto	Cómo se accederá a la ayuda estática	<i>MainMenu</i> almacena la entrada del menú para la ayuda

Tabla 10.8 Resumen de los cambios para incorporar *Multilevel Help*

PARTE III: EVALUACIÓN Y CONCLUSIONES

Capítulo 11

Evaluación de Viabilidad

El objetivo de este capítulo es el de explicar cómo se modificaría el proceso de educación entre el analista y el usuario dentro de la TTM elegida para incorporar los mecanismos de usabilidad. Esto permitirá hacer un estudio sobre la viabilidad de incorporar los mecanismos de usabilidad a una TTM. Como ejemplo de TTM, se ha elegido OO-Method.

Actualmente, el proceso de desarrollo de sistemas en OO-Method consiste en construir sus modelos de una manera ordenada. En primer lugar se crea una primera versión del Modelo de Objetos. Una vez se tiene una versión del Modelo de Objetos con atributos y servicios, se construyen el Modelo Dinámico y Funcional. Finalmente, se construye el Modelo de Presentación en sus dos vistas, Modelo de Interacción Abstracto y Modelo de Interacción Concreto. La construcción de estos modelos no tiene porqué hacerse de manera secuencial, sino que se puede hacer en espiral, es decir, en sucesivas iteraciones. Esta sección explica cómo desarrollar aplicaciones OO-Method incorporando los cambios propuestos en cada uno de los modelos OO-Method. Para ello vamos a tomar como ejemplo el desarrollo del sistema de alquiler de coches (Figura 4.5).

En las siguientes secciones comentan los cambios que afectan al: Modelo de Objetos, Modelo de Interacción Abstracto, Modelo de Interacción Concreto. El Modelo Dinámico y Funcional no se ven afectados por la incorporación de mecanismos de usabilidad, por lo que no se contemplan en este capítulo.

11.1 Construcción del Modelo de Objetos

El Modelo de Objetos es el primer modelo que se construye siguiendo la metodología OO-Method. Los pasos que debe seguir el analista para la construcción de este modelo actualmente son los siguientes:

1. Creación de las clases del sistema
2. Para cada clase se definen sus atributos
3. Para cada clase se definen sus servicios
4. Para cada servicio se definen sus argumentos de entrada
5. Finalmente se definen las precondiciones y las restricciones de integridad:
 - Las precondiciones se utilizan para asegurar que los servicios se ejecutarán sólo bajo ciertas condiciones asociadas.
 - Las restricciones de integridad se utilizan para evitar que algunos atributos tomen valores no correctos.

El modelado de las formas de uso se hace embebido durante los cinco pasos que componen la construcción del Modelo de Objetos. A continuación se explica cómo modelar cada una de las formas de uso ordenadas según el instante del tiempo en el cual se deben tratar. Para ilustrar cómo se modela cada una de las formas de uso en OO-Method, se han utilizado prototipos.

Las formas de uso que afectan al modelo de objetos y que se discuten en esta sección son las siguientes:

- Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)
- Deshacer cambios (WU_GU1)

- Rehacer cambios (WU_GU2)
- Cancelar durante la ejecución (WU_AO1)
- Valores por defecto (WU_STE3)
- Mensaje de aviso (WU_W1)

11.1.1 Modelado de WU_SSF1

Esta sección va a explicar cómo modelar en OO-Method la forma de uso *Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)* para cada servicio dentro del Modelo de Objetos. Una vez creado el servicio, el analista puede especificar un mensaje a mostrar al usuario cuando el servicio se ejecute correctamente, o cuando no se pueda ejecutar por un fallo en la transición de estados. En caso de que el analista no especifique estos mensajes, el compilador de modelos se encargará de crear uno genérico para todos los servicios del sistema.

La Figura 11.1 muestra un prototipo de cómo quedaría la ventana del Modelo de Objetos desde la que se modelaría WU_SSF1. El analista debe seleccionar el servicio sobre el que se van a definir los mensajes, y posteriormente se debe pulsar el botón *Opciones avanzadas* (remarcado en la figura en un recuadro). En la Figura 11.1 se ha tomado como ejemplo el servicio *crear alquiler*. Tras pulsar el botón *Opciones avanzadas*, se muestra una ventana emergente como la de la Figura 11.2 en la que el analista debe introducir los dos mensajes de texto. Lo escrito en el campo *Success message* se mostrará cuando el servicio se ejecute correctamente, mientras que lo escrito en el campo *Failure message* se mostrará cuando el usuario solicite la ejecución del servicio en un estado del objeto desde el que no se puede ejecutar. Tras escribir ambos mensajes se acaba la parte de modelado de WU_SSF1 en el Modelo de Objetos. La imagen de la ventana de *Opciones avanzadas* (Figura 11.2) es sólo una parte de dicha ventana. Esta ventana incluye más funcionalidades que se explicarán en las siguientes secciones.

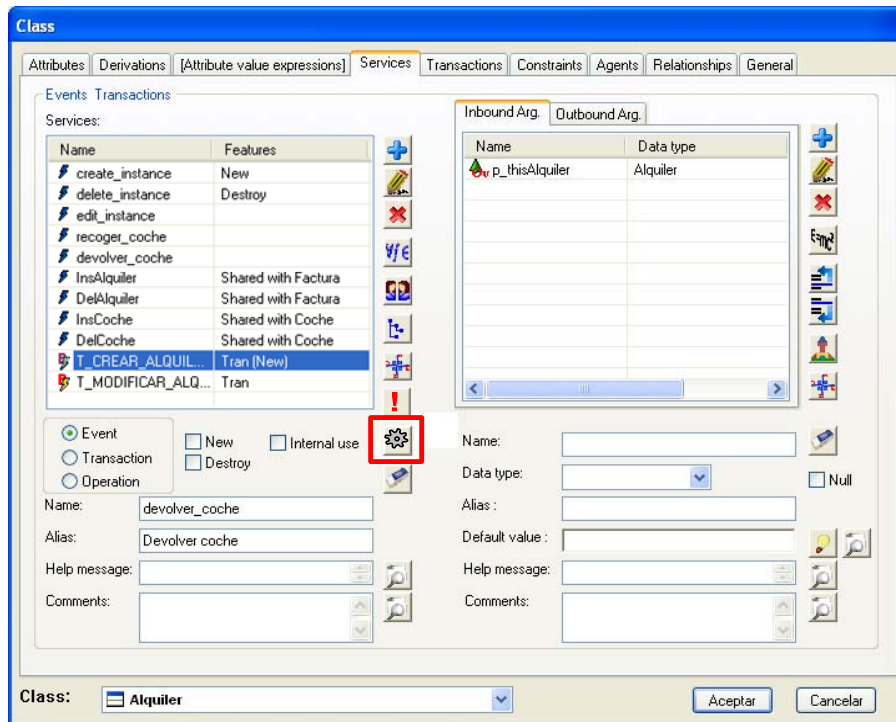


Figura 11.1 Modelo de Objetos para modelar SSF1 (1)

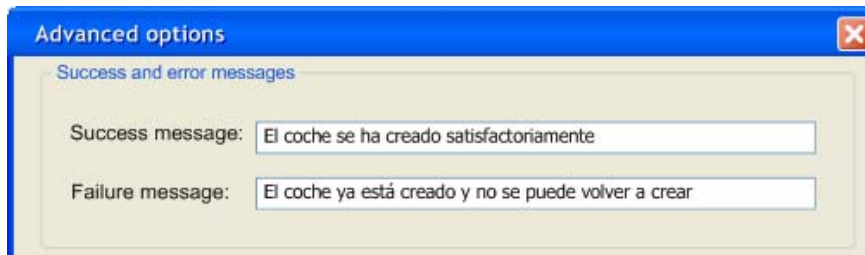


Figura 11.2 Modelo de Objetos para modelar SSF1 (2)

11.1.2 Modelado de WU_GU1 y WU_GU2

Esta sección explica cómo modelar las formas de uso *Deshacer cambios* (*WU_GU1*) y *Rehacer cambios* (*WU_GU2*) dentro del Modelo de Objetos, ya que la funcionalidad de ambas está relacionada. Para acceder al modelado de ambas formas de uso se debería pulsar el botón de *Opciones avanzadas* resaltado en la Figura 11.1. Por lo tanto, *WU_GU1* y *WU_GU2*

se pueden modelar al mismo tiempo que se modele WU_SSF1, es decir, tras la definición del servicio.

La ventana que se abre tras pulsar el botón *Opciones avanzadas* es como la mostrada en la Figura 11.3. Es importante resaltar que esta ventana es la misma que la representada en la Figura 11.2, pero visualizando no sólo la porción de ventana desde la que se modela WU_SSF1, sino también la porción desde la que se modelan WU_GU1 y WU_GU2.

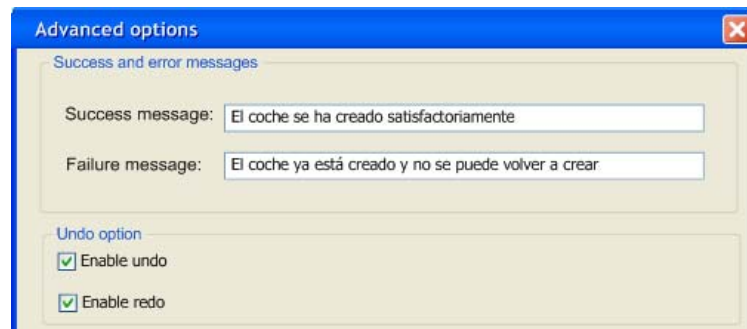


Figura 11.3 Modelo de Objetos para modelar WU_GU1 y WU_GU2

Para habilitar la funcionalidad de WU_GU1 y WU_GU2, el analista sólo tendría que marcar los checkboxes *Enable undo* y *Enable redo* respectivamente. En el ejemplo de la Figura 11.3 se habría habilitado la funcionalidad de deshacer y rehacer para el servicio *crear coche*.

11.1.3 Modelado de WU_AO1

Esta sección explica cómo modelar la forma de uso *Cancelar durante la ejecución (WU_AO1)* en el Modelo de Objetos de OO-Method. Para acceder al modelado de WU_AO1, se debería pulsar el botón de *Opciones avanzadas* resaltado en la Figura 11.1. Por lo tanto, WU_AO1 se puede modelar al mismo tiempo que se modele WU_SSF1, WU_GU1 y WU_GU2, es decir, tras definir el servicio.

La ventana que se abriría tras pulsar el botón *Opciones avanzadas* es como la mostrada en la Figura 11.4. Esta figura es la misma ventana que la representada en la Figura 11.2 para WU_SSF1 y en la Figura 11.3 para

WU_GU1 y WU_GU2. En la Figura 11.4 se muestra la ventana *Opciones avanzadas* con toda su funcionalidad.

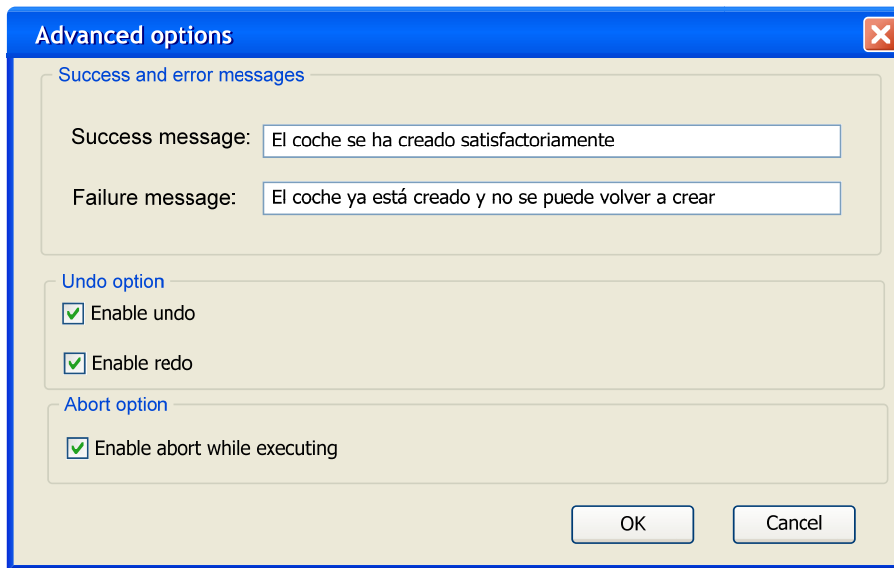


Figura 11.4 Modelo de Objetos para modelar WU_AO1

Para habilitar que el servicio *crear coche* se pueda cancelar a mitad ejecución, bastaría con marcar el checkbox etiquetado como *Enable abort while executing*. El resto de formas de uso que afectan al Modelo de Objetos se modelan desde ventanas distintas a *Opciones avanzadas*.

11.1.4 Modelado de WU_STE3

Esta sección explica cómo modelar la forma de uso *Valores por defecto* (WU_STE3). El modelado de esta forma de uso ya está soportado actualmente por OO-Method y por la herramienta OlivaNOVA, por lo tanto las capturas de ventana son totalmente funcionales.

El modelado de WU_STE3 se hace en la definición de los argumentos de entrada relacionados con los servicios. Una vez definidos los servicios, el analista debe definir los argumentos necesarios para la ejecución de dicho servicio. La definición de los argumentos de entrada se hace desde una ventana como la mostrada en la Figura 11.5. En dicha figura se puede apreciar cómo además de poder especificar el nombre del argumento, su

tipo y su alias, también se puede especificar un valor por defecto (campo *Default value* resaltado).

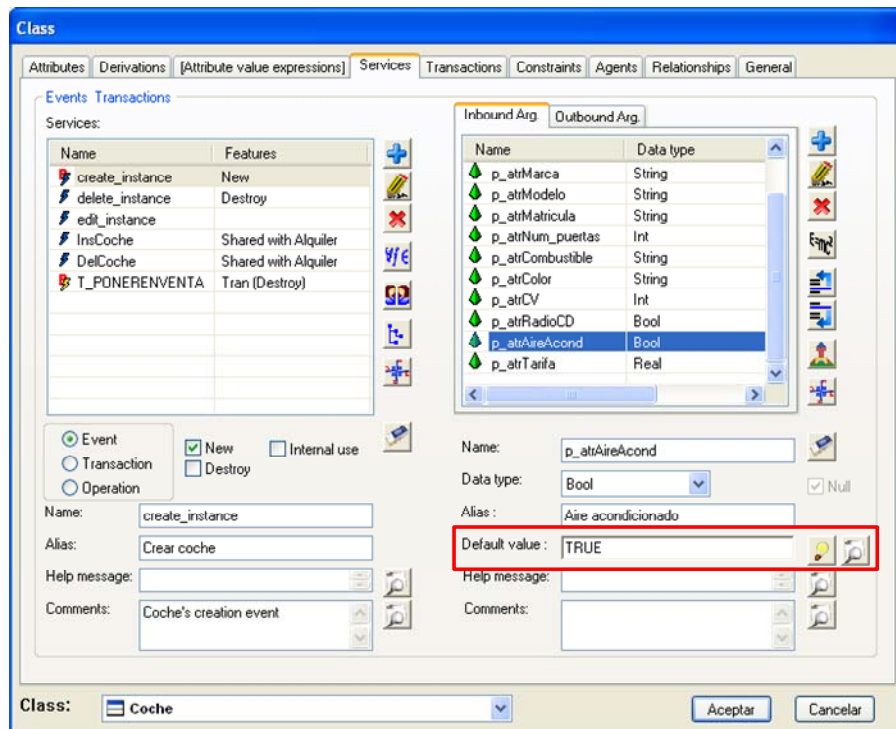


Figura 11.5 Modelo de Objetos para modelar WU_STE3

Como ejemplo, en la Figura 11.5 se ha representado cómo se definiría el valor por defecto para el argumento de entrada *Aire acondicionado* del servicio *crear coche* de la clase *Coche*. El valor por defecto que se ha definido para este argumento es el de *Verdadero*.

11.1.5 Modelado de WU_W1

Esta sección explica cómo modelar la forma de uso *Mensaje de aviso* (*WU_W1*) en el Modelo de Objetos de OO-Method. Una vez definidos los servicios y sus argumentos de entrada, el siguiente paso es el de definir las precondiciones y las restricciones de integridad. Es en este momento cuando se debe modelar la forma de uso *WU_W1*.

La Figura 11.6 muestra una ventana con el prototipo para implementar WU_W1. En primer lugar, se debe pulsar el botón de *Mensaje de aviso*, resaltado en la Figura 11.6. Tras pulsar dicho botón, se muestra una ventana emergente similar a la de la Figura 11.7, en la que el analista puede especificar cuándo se visualizará el mensaje de aviso y el contenido de dicho mensaje. En el campo *Fórmula*, el analista debe especificar la condición bajo la cual se mostrará el mensaje de aviso, mientras que en el campo *Warning message*, el analista debe especificar el contenido del mensaje de aviso que se mostrará al usuario.

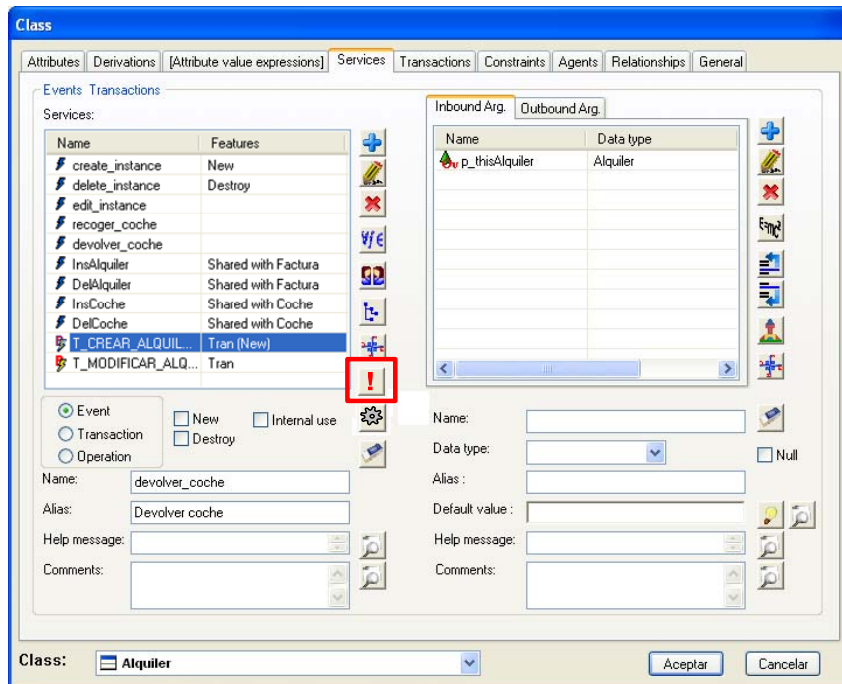


Figura 11.6 Modelo de Objetos para modelar WU_W1(1)

En la Figura 11.6 y Figura 11.7 se muestra a modo de ejemplo un mensaje de aviso que se mostraría al ejecutar el servicio *crear alquiler* de la clase *Alquiler*. El mensaje de aviso se mostrará al usuario en el caso de que el alquiler se haga por más de un mes.

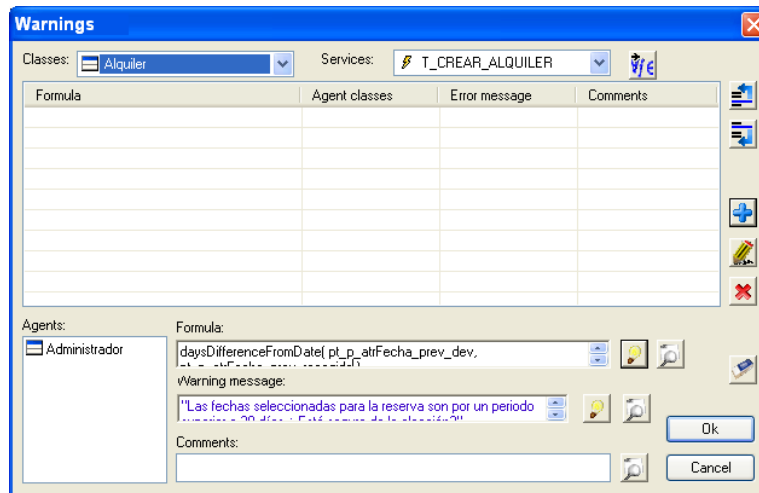


Figura 11.7 Modelo de Objetos para modelar WU_W1(2)

De esta manera se modelarían en el Modelo de Objetos todas las formas de uso incluidas en esta sección. Es importante resaltar que muchas de estas formas de uso se modelan también en otros modelos de OO-Method, y no sólo en el Modelo de Objetos. En las siguientes secciones se tratará el modelado de las formas de uso en otros modelos.

11.2 Construcción del Modelo de Interacción Abstracto

Una vez construido el Modelo de Objetos, los siguientes modelos que se construyen son el Modelo Dinámico y el Modelo Funcional. Estos modelos no se ven afectados por la incorporación de las formas de uso, por lo que no se tratan en este capítulo.

Una vez construidos el Modelo Dinámico y Funcional, el siguiente modelo a utilizar para representar el sistema es el Modelo de Interacción Abstracto. Los pasos que debe seguir el analista actualmente para la construcción del Modelo de Interacción Abstracto son los siguientes:

1. Definición de Unidades de Interacción

2. Definición de Patrones Elementales
3. Definición de Patrones Auxiliares (subconjunto de Patrones Elementales formado por el patrón de Introducción y el de Selección definida)
4. Definición del Árbol de Jerarquía de Acciones (menú del sistema)

El modelado de las formas de uso que afectan al Modelo de Interacción Abstracto se hace embebido dentro de estos cuatro pasos. A continuación se explica, en base a prototipos, cómo modelar cada una de las formas de uso ordenadas según el instante del tiempo en el cual se deben modelar.

Las formas de uso que afectan al modelo de interacción abstracto y que se discuten en esta sección son las siguientes:

- Definir un asistente (WU_SBS1)
- Mostrar progreso de la ejecución (WU_PF1)
- Mostrar el estado de la información (WU_SSF2)
- Mostrar el estado de las acciones (WU_SSF3)
- Definición de máscaras (WU_STE2)
- Ayuda dinámica (WU_MH1)
- Deshacer cambios (WU_GU1)
- Rehacer cambios (WU_GU2)
- Ayuda estática (WU_MH2)
- Definición de favoritos (WU_F1)

11.2.1 Modelado de WU_SBS1

Esta sección va a explicar cómo modelar en OO-Method la forma de uso *Definir un asistente (WU_SBS1)* dentro del Modelo de Interacción Abstracto.

La forma de uso WU_SBS1 se modela cuando se construyen las distintas Unidades de Interacción del sistema, es decir, cuando se construyen las Unidades de Interacción de Servicio, Población, Instancia y Maestro/Detalle. Se debe añadir al Modelo de Interacción Abstracto un nuevo tipo de UI, el Wizard UI. Esta Unidad de Interacción se utilizará sólo para modelar un asistente. Para ello, el analista debería ir a la ventana desde la que se crean nuevas Wizard UI, seleccionar la clase que contiene el servicio sobre el que desea definir el asistente y pulsar el botón de Crear (resaltado en rojo en la Figura 11.8). De esta manera se abre la ventana desde la que se puede modelar el asistente.

La funcionalidad de WU_SBS1 se modela en dos pestañas: la de definición del asistente (*Definition*) y la de edición de los pasos (*Step edition*). La Figura 11.8 muestra un prototipo de ventana para la pestaña *Definition*. El analista debería especificar en el campo *Service*, el servicio que se ejecutará por medio del asistente. En el campo *Steps*, se listan todos los pasos que forman el asistente. Cada vez que el analista pulsa el botón de crear (símbolo +) o el botón de modificar (lápiz), se abre la pestaña *Step edition*. En el ejemplo de la Figura 11.8 se han definido siete pasos para el servicio *crear alquiler* de la clase *alquiler*.

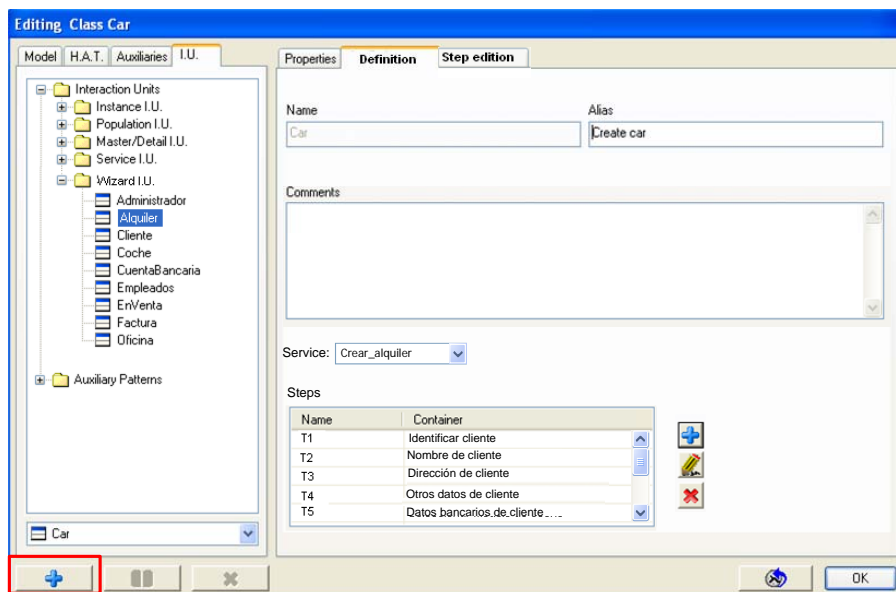


Figura 11.8 Modelo de Interacción Abstracto para modelar WU_SBS1 (1)

La Figura 11.9 presenta un prototipo de ventana para editar un paso concreto del asistente (pestaña *Task edition*). En primer lugar, el analista debe especificar los atributos que se muestran en el paso. Los atributos que se pueden seleccionar son los que existen en cualquiera de las clases definidas. Para indicar qué atributos se van a mostrar en el paso, el analista debe seleccionar un atributo de la ventana izquierda y pulsar la flecha para almacenar el atributo en la ventana de la derecha. La flecha en sentido contrario se puede utilizar para borrar atributos que ya han sido seleccionados previamente. Además, existen flechas para modificar el orden de aparición de los atributos en la interfaz. Después, el analista puede introducir un identificador para el paso (campo *Step id*), un nombre que será el título del paso (campo *Step name*), y una breve descripción que explique al usuario lo que debe hacer en el paso actual. En el ejemplo de la Figura 11.9 se mostraría en el primer paso del asistente los atributos: *Nombre*, *Apellidos*, *DNI* y *Es_nuevo*. El título para este paso es *Identificar cliente* y le acompaña una breve descripción sobre su funcionalidad.

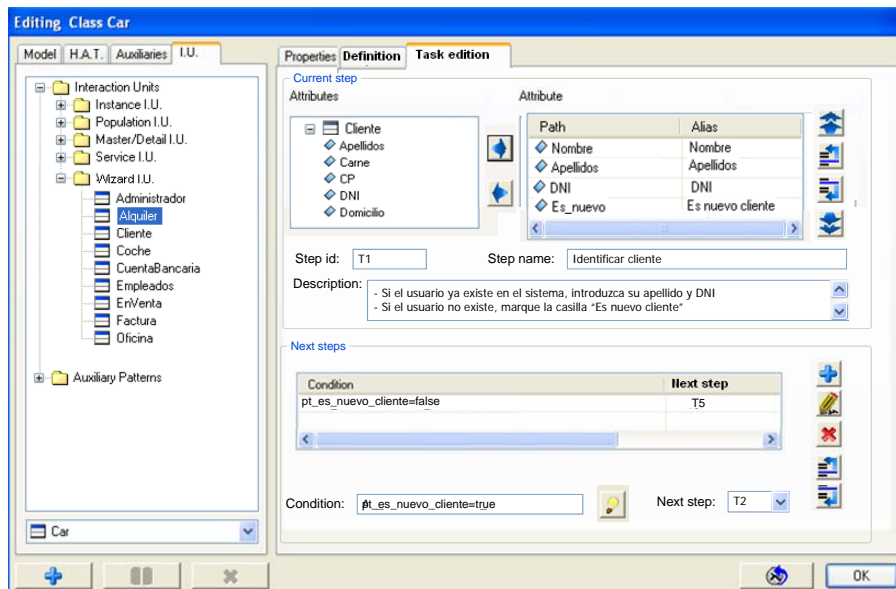


Figura 11.9 Modelo de Interacción Abstracto para modelar WU_SBS1 (2)

Una vez definido el paso actual, el analista debe especificar cuál será el flujo de ejecución entre los pasos, es decir, cuál será el siguiente paso en mostrarse al usuario. Para ello introduce condiciones que dependiendo del

valor que tomen, guiarán al usuario a uno u otro paso. El analista debe introducir la fórmula de la condición en el campo *Condition* y el próximo paso al que se accederá en caso de que la condición se cumpla en el campo *Next step*. Una vez definida la condición y cuál será el siguiente paso para esa condición, se debe pulsar el botón *Añadir* (representado con el icono +) para guardar los cambios. Si el siguiente paso no está sujeto a ninguna condición, se puede dejar el campo *Condition* vacío. En el ejemplo de la Figura 11.9, el flujo de ejecución seguirá en el paso cinco si el cliente ya existe previamente y en el paso dos si el usuario es nuevo.

11.2.2 Modelado de WU_PF1

Esta sección explica cómo modelar la forma de uso *Mostrar progreso de la ejecución (WU_PF1)* en el Modelo de Interacción Abstracto de OO-Method. Esta forma de uso se modela cuando se especifican las Unidades de Interacción de Servicio, como una propiedad más de estas UIs.

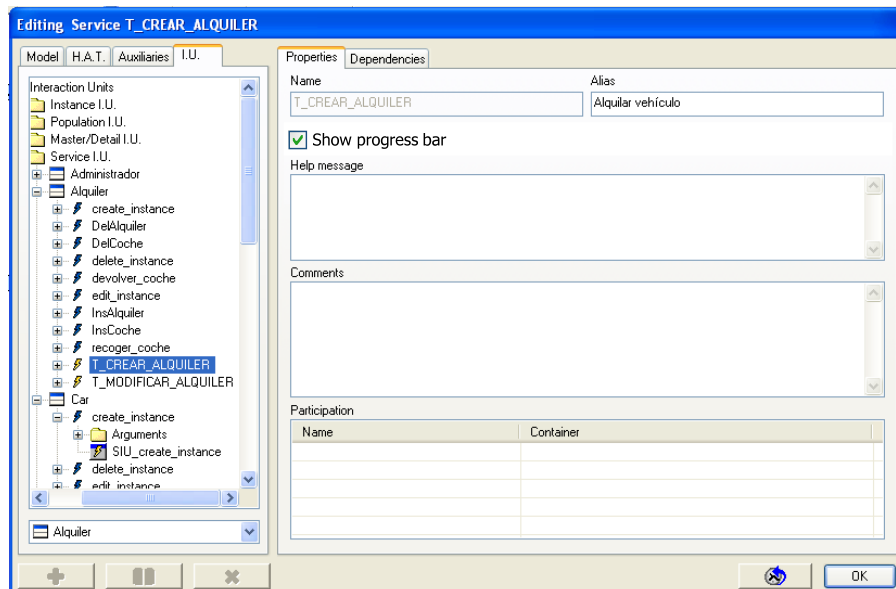


Figura 11.10 Modelo de Interacción Abstracto para modelar WU_PF1

La Figura 11.10 muestra un prototipo de ventana desde la que se podría modelar esta forma de uso. Para indicar que un servicio va a mostrar una barra de progreso, se debe seleccionar la UI de Servicio sobre la que se va

a mostrar la barra de progreso. En la pestaña *Properties* de la UI de Servicio, basta con marcar la casilla *Show progress bar*. En el ejemplo de la Figura 11.9 se ha habilitado la barra de progreso para el servicio *crear alquiler*.

Esta es la última de las formas de uso que afecta a la definición de Unidades de Interacción. Una vez definidas las Unidades de Interacción, el analista de OO-Method debe definir los Patrones Elementales dentro de cada una de las Unidades de Interacción creadas. A continuación se van a explicar los cambios que afectan a la definición de Patrones Elementales dentro del Modelo de Interacción Abstracto.

11.2.3 Modelado de WU_SSF2 y WU_SSF3

Esta sección explica cómo modelar la forma de uso *Mostrar el estado de la información (WU_SSF2)* y la forma de uso *Mostrar el estado de las acciones (WU_SSF3)* dentro del Modelo de Interacción Abstracto. Ambas formas de uso se modelan dentro del Patrón Elemental de Navegación, una vez se han definido las Unidades de Interacción.

En primer lugar se va a explicar cómo modelar WU_SSF2 mediante la Figura 11.11. Este prototipo de ventana representa la edición de una navegación previamente creada. El analista puede introducir el alias estático en el campo *Static alias* y el alias dinámico en el campo *Dynamic alias*. En el ejemplo de la Figura 11.11 se muestra una navegación hacia instancias de la clase *Factura* desde una UI de Población de la clase *Alquileres*. Esta navegación tiene como alias estático la cadena *Facturas* y como alias dinámico el número de facturas del alquiler seleccionado.

Por lo que respecta a la forma de uso WU_SSF3, se puede modelar con el campo *Disable navigation if the navigation is empty* de la Figura 11.11. El analista debe marcar esta casilla en el caso de que quiera deshabilitar la navegación si el contexto destino no tiene instancias. Al marcar la casilla de la Figura 11.11, se deshabilitará la navegación en la interfaz generada en caso de que no haya facturas relacionadas con la instancia seleccionada de la clase *Alquileres*.

Editing Navigation Element

Properties

Name: Static alias:

Dynamic alias:

Help message: Comments:

Navigational filtering

Filtering condition:

Comments:

Participation

Name	Container

Disable navigation if the destination is empty

Figura 11.11 Modelo de Interacción Abstracto para modelar WU_SSF3

WU_SSF2 y WU_SSF3 son las únicas formas de uso que afectan a los Patrones Elementales del Modelo de Interacción Abstracto. Una vez definidos los Patrones Elementales, el analista de OO-Method debe definir los Patrones Auxiliares. A continuación se va a explicar los cambios que implican las formas de uso sobre los Patrones Auxiliares del Modelo de Interacción Abstracto.

11.2.4 Modelado de WU_STE2

Esta sección explica cómo modelar la forma de uso *Definición de máscaras* (WU_STE2) en el Modelo de Interacción Abstracto de OO-Method. Esta forma de uso se modela dentro del Patrón Auxiliar de Introducción.

Esta forma de uso ya está soportada por OO-Method y está implementado en la herramienta OlivaNOVA. La Figura 11.12 muestra la ventana de OlivaNOVA desde la que se modelan las máscaras hoy en día. En esta figura se está modelando, a modo de ejemplo, una máscara para el campo código postal del cliente. La máscara se utiliza en este caso para asegurar que el código postal es una cadena compuesta por 5 dígitos.

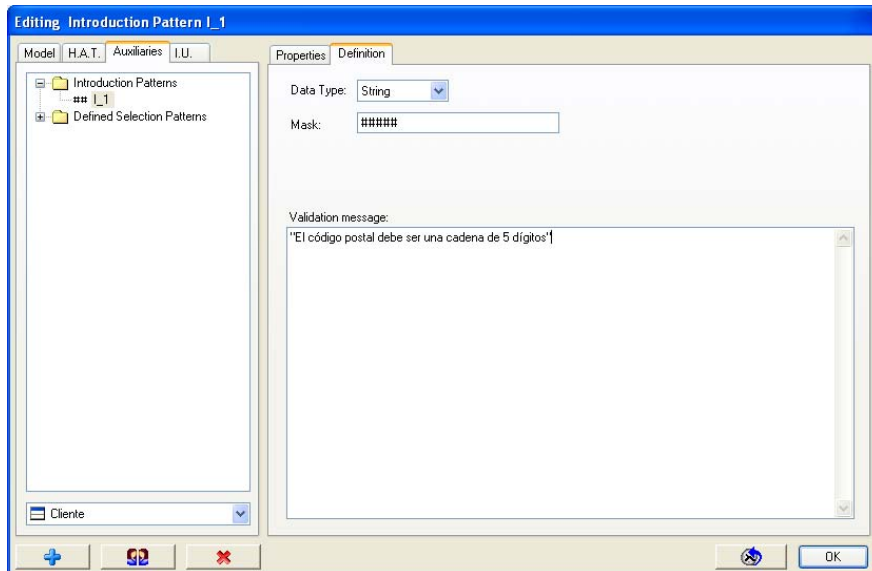


Figura 11.12 Modelo de Interacción Abstracto para modelar WU_STE2

11.2.5 Modelado de WU_MH1

Esta sección explica cómo modelar la forma de uso *Ayuda dinámica* (*WU_MH1*) en el Modelo de Interacción Abstracto de OO-Method. Esta forma de uso se modela dentro del Patrón Auxiliar de Introducción.

La Figura 11.13 muestra un prototipo de ventana desde la que modelar esta forma de uso. Al seleccionar la carpeta *Dynamic help*, el analista accede a las opciones relacionadas con la configuración de la ayuda dinámica. Estas opciones son las mostradas en la Figura 11.13. Marcando las distintas casillas de la figura se habilitarán los siguientes elementos en el sistema desarrollado: la ayuda del sistema en general; la ayuda de los elementos de entrada de datos; la ayuda de las variables de filtro; la ayuda de los criterios

de ordenación; la ayuda de los elementos del conjunto de visualización; la ayuda de las acciones; la ayuda de las navegaciones. Las casillas que se desmarquen no mostrarán su ayuda dinámica asociada. En el caso de ejemplo de la Figura 11.13, están todos los mensajes de ayuda habilitados.

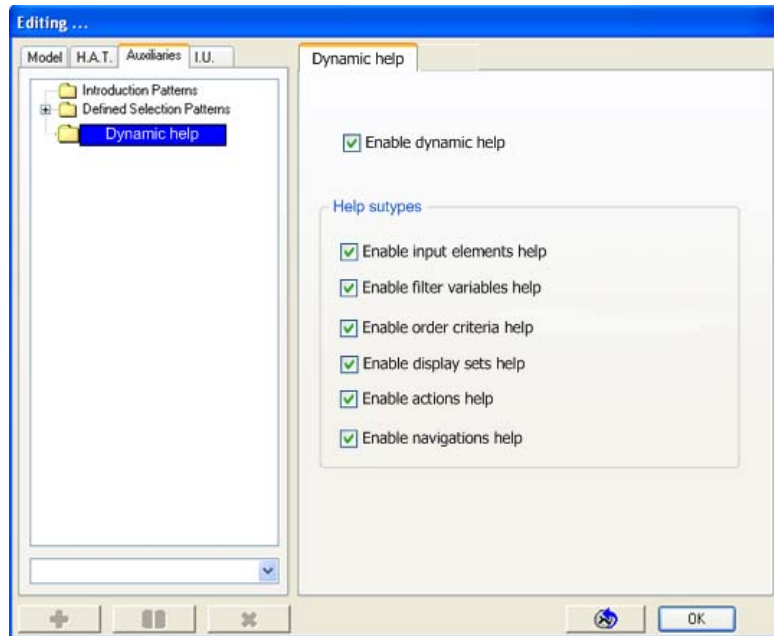


Figura 11.13 Modelo de Interacción Abstracto para modelar WU_MH1

Esta es la última de las formas de uso que afecta a la definición de Patrones Auxiliares. Por último, una vez definidas las Unidades de Interacción, los Patrones Elementales y los Auxiliares, el último paso es la definición del Árbol de Jerarquía de Acciones (menú del sistema). A continuación, se van a explicar los cambios provocados por las formas de uso sobre el Árbol de Jerarquía de Acciones.

11.2.6 Modelado de GU1 y GU2

Esta sección explica cómo modelar la forma de uso *Des hacer cambios* (*WU_GU1*) y *Re hacer cambios* (*WU_GU2*) en el Modelo de Interacción Abstracto de OO-Method. Ambas formas de uso se modelan dentro del Árbol de Jerarquía de Acciones.

La Figura 11.14 muestra una captura de ventana real que se utiliza actualmente para modelar el Árbol de Jerarquía de Acciones en OO-Method. En la sección *Actions Hierarchy* se muestra cómo quedaría la jerarquía del menú para acceder a la funcionalidad del sistema.

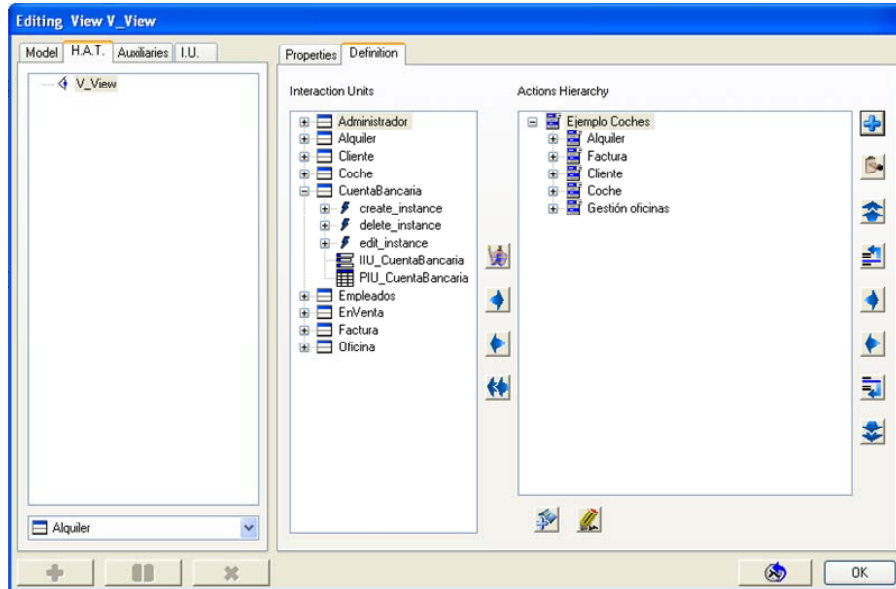


Figura 11.14 Modelado del Árbol de Jerarquía de Acciones

El analista puede añadir nuevos elementos a las entradas del menú. Para añadir nuevos elementos, debe pulsar el botón *Añadir* (+) y se muestra una ventana como la de la Figura 11.15. Dicha figura muestra los distintos elementos que se pueden agregar a una entrada del menú. Como novedad con respecto a lo que ya existe hoy en día en OO-Method, se propone la incorporación del elemento *Common functionalities*. Este tipo de entrada del menú es para designar todas aquellas funcionalidades que son útiles para cualquier sistema independientemente de su lógica de negocio, por ejemplo, la función de imprimir. En este tipo de entrada de menú se agregarían las funcionalidades deshacer (WU_GU1) y rehacer cambios (WU_GU2).

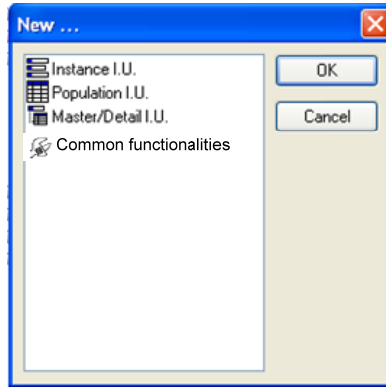


Figura 11.15 Añadir un nuevo elemento al Árbol de Jerarquía de Acciones

Una vez indicado que se va a agregar una nueva entrada de menú para una funcionalidad común, el siguiente paso es el de darle un alias y una combinación de teclas de acceso rápido. La Figura 11.16 muestra el prototipo de una ventana desde la que se define el alias que verá el usuario en la aplicación generada y la secuencia de teclas rápidas para acceder también por teclado. En el ejemplo de la Figura 11.16 se está definiendo el alias *Deshacer* y CTRL+Z como combinación de teclas de acceso rápido.

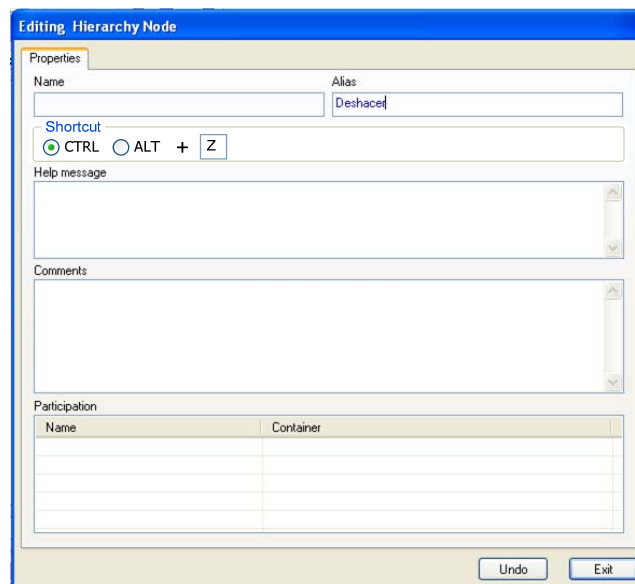


Figura 11.16 Modelado del acceso a las funcionalidades de WU_GU1 y WU_GU2 a partir del menú del sistema (1)

La Figura 11.17 muestra, cómo queda el Árbol de Jerarquía de Acciones una vez el analista ha especificado que la funcionalidad de deshacer y rehacer se va a acceder desde el menú *Edición* de la aplicación generada.

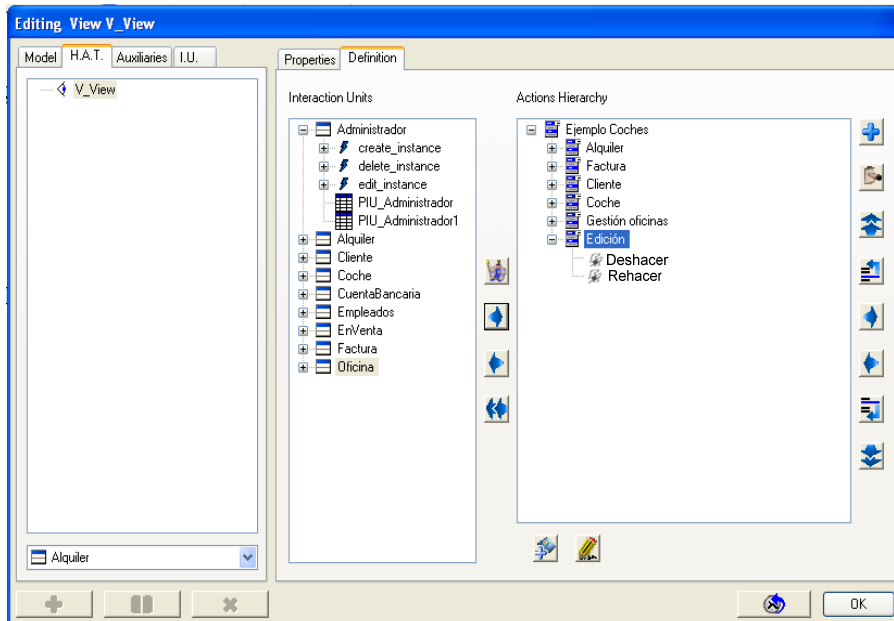


Figura 11.17 Modelado del acceso a las funcionalidades de WU_GU1 y WU_GU2 a partir del menú del sistema (2)

11.2.7 Modelado de WU_MH2

Esta sección explica cómo modelar la forma de uso *Ayuda estática* (WU_MH2) en el Modelo de Interacción Abstracto de OO-Method. Esta forma de uso se modela dentro del Árbol de Jerarquía de Acciones.

Para introducir una nueva entrada en el menú del sistema que acceda a la ayuda estática, se debe utilizar un Árbol de Jerarquía de Acciones como el de la Figura 11.14. En este árbol se debe añadir una nueva entrada del tipo *Common functionalities* (Figura 11.15), ya que la ayuda estática es una funcionalidad que podría estar en cualquier tipo de sistema. Una vez introducida la nueva entrada del menú, en una ventana como la mostrada en la Figura 11.18 se debe indicar un alias para la nueva entrada y una

combinación de teclas de acceso rápido. En el ejemplo de la Figura 11.14 se muestra como alias *Ayuda* y como teclas de acceso rápido CTRL+F1.

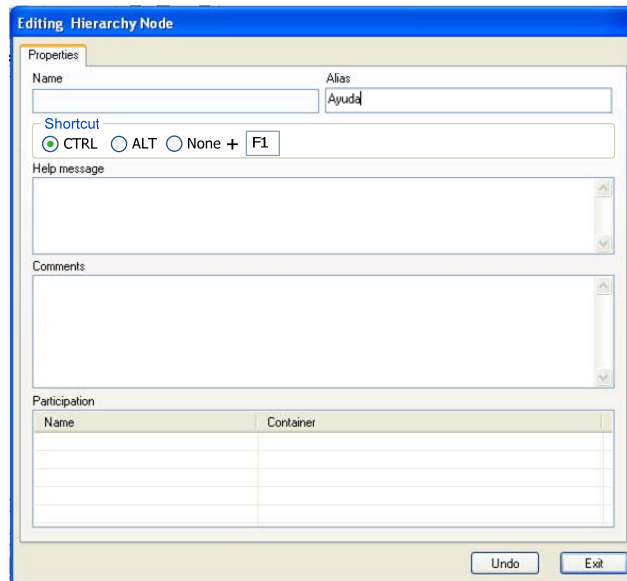


Figura 11.18 Modelado del acceso a la funcionalidad WU_MH2

11.2.8 Modelado de WU_F1

Esta sección explica cómo modelar la forma de uso *Definición de favoritos* (*WU_F1*) en el Modelo de Interacción Abstracto de OO-Method. Concretamente, se modela cómo se accederá a la lista de favoritos dentro del sistema. Para ello hay que aplicar cambios al Árbol de Jerarquía de Acciones.

Para introducir una nueva entrada en el menú del sistema que acceda a la lista de favoritos, se debe utilizar un Árbol de Jerarquía de Acciones como el de la Figura 11.14. En este árbol se ha de añadir una nueva entrada del tipo *Common functionalities* (Figura 11.15). Una vez introducida la nueva entrada del menú, en una ventana como la mostrada en la Figura 11.19 se indica un alias para la nueva entrada y una combinación de teclas de acceso rápido. En el ejemplo de la Figura 11.19, se muestra como alias *Favoritos* y como teclas de acceso rápido CTRL+F.

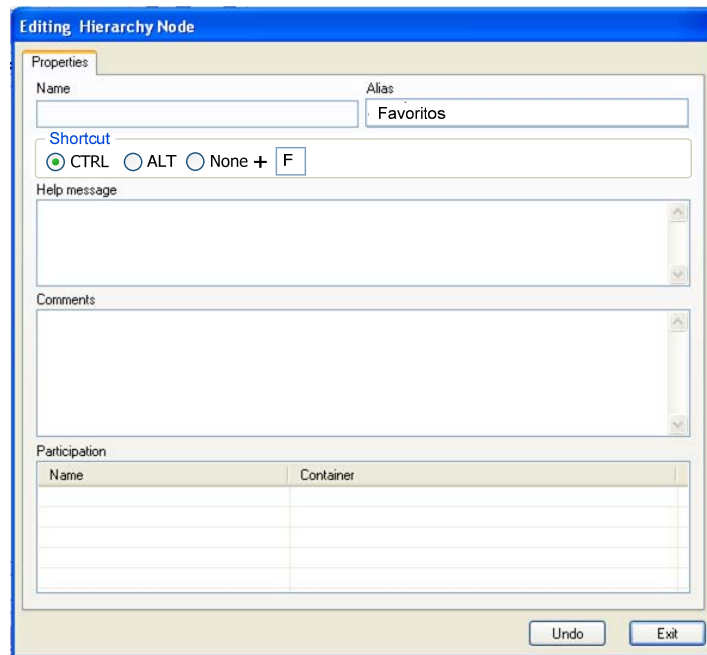


Figura 11.19 Modelado del acceso a la funcionalidad WU_F1

Con esta forma de uso se termina la explicación de cómo modelar las formas de uso en el Modelo de Interacción Abstracto. A continuación se va a explicar cómo modelar las formas de uso en la última vista que debe modelar el analista de OO-Method: el Modelo de Interacción Concreto.

11.3 Construcción del Modelo de Interacción Concreto

El Modelo de Interacción Concreto es el último en la secuencia de modelos con los que debe trabajar el analista. En este modelo se especifica cómo se visualizarán los distintos elementos que componen la interfaz del sistema. La secuencia de acciones que debe realizar el analista en este modelo es:

1. Especificar los aspectos de interacción concretos para las Unidades de Interacción

2. Especificar los aspectos de interacción concretos para los Patrones Elementales

El modelado de las formas de uso que afectan al Modelo de Interacción Concreto se hace dentro de estos 2 pasos. Las formas de uso que afectan al modelo de interacción concreto y que se discuten en esta sección son las siguientes:

- Definir un asistente (WU_SBS1)
- Especificar el tipo de visualización del campo de entrada (WU_STE1)
- Preferencias en el aspecto visual (WU_P1)
- Distribución de elementos (WU_POS1)
- Cancelar durante la ejecución (WU_AO1)
- Salir de un escenario (WU_AO2)
- Ayuda dinámica (WU_MH1)
- Mostrar progreso de la ejecución (WU_W1)
- Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)
- Mostrar el estado de la información (WU_SSF2)
- Mensaje de aviso (WU_W1)
- Definición de favoritos (WU_F1)

11.3.1 Modelado de WU_SBS1

Esta sección va a explicar cómo modelar en OO-Method la forma de uso *Definir un asistente (WU_SBS1)* dentro del Modelo de Interacción Concreto. WU_SBS1 se modela dentro de la sección donde se modelan las Unidades de Interacción, en concreto, se modela dentro de las Wizard UI. Esta forma de uso sólo incluye en el Modelo de Interacción Concreto la capacidad de

decidir cómo se distribuyen los distintos elementos de cada uno de los pasos del asistente en las ventanas.

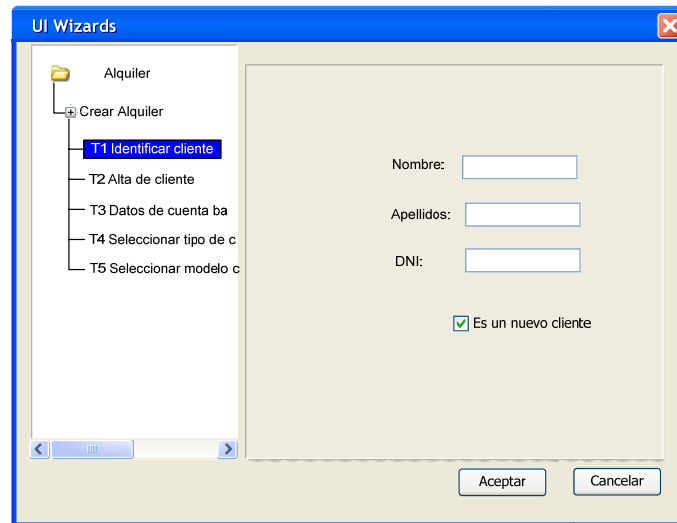


Figura 11.20 Modelo de Interacción Concreto para modelar WU_SBS1

La Figura 11.20 muestra un prototipo de una ventana que modelaría la posición de los campos de entrada de datos para modelar la Figura 6.1. En el árbol de la izquierda, se elegiría el paso sobre el que se va a modelar la posición de los campos. En el caso de ejemplo, se corresponde con el paso *Identificar cliente*, que pertenece al servicio *crear alquiler*, que a su vez está dentro de la clase *Alquiler*. El analista sólo tendría que mover los componentes a la posición deseada en la porción derecha de la ventana.

11.3.2 Modelado de WU_STE1

Esta sección explica cómo modelar la forma de uso *Especificar el tipo de visualización del campo de entrada (WU_STE1)* en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan las UIs de Servicio.

La Figura 11.21 presenta un prototipo de ventana donde indicar el tipo de campo con el que se representarán los argumentos de entrada de las interfaces de servicio generadas. En la parte izquierda de la ventana aparecen las distintas UIs de Servicio agrupadas por clases del Modelo de

Objetos. Dentro de Cada UI de Servicio se pueden observar los argumentos. En el ejemplo se muestran los tipos de argumentos que se han asignado a la UI de Servicio *Alta de cliente*. Para cambiar el tipo de argumento, basta con seleccionar el argumento deseado y en la porción derecha de la ventana se puede cambiar el tipo. En el ejemplo se ha elegido la introducción del argumento *Provincia* mediante una lista desplegable.

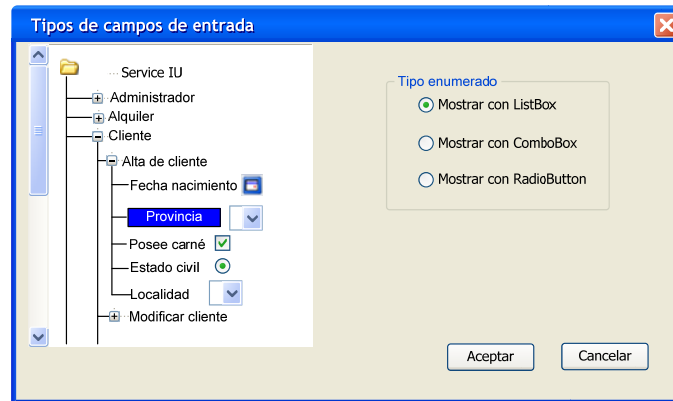


Figura 11.21 Modelo de Interacción Concreto para modelar WU_STE1

11.3.3 Modelado de WU_P1

Esta sección explica cómo modelar la forma de uso *Preferencias en el aspecto visual (WU_P1)* en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan todas las Unidades de Interacción.

Esta forma de uso incluye un gran número de primitivas conceptuales para habilitar o deshabilitar la posibilidad de que el usuario modifique el aspecto visual de las interfaces en base a sus preferencias. Estas primitivas se han agrupado en cuatro tipos distintos dependiendo del tipo de personalización que habiliten o deshabiliten: texto, iconos, elementos y colores. Cada uno de estos tipos da lugar a una pestaña en la interfaz donde se modela WU_P1.

La Figura 11.22 muestra un prototipo para modelar la habilitación para personalizar el texto de las interfaces. En la porción izquierda de la ventana se elige mediante un menú en forma de árbol la UI sobre la que se va a

modificar la capacidad de personalización. En la porción derecha de la ventana se visualiza la funcionalidad para habilitar o deshabilitar la personalización del texto en la Unidad de Interacción generada. Se podría habilitar la personalización del tamaño y del estilo del texto. En el caso de ejemplo de la Figura 11.22, ambas personalizaciones están habilitadas.

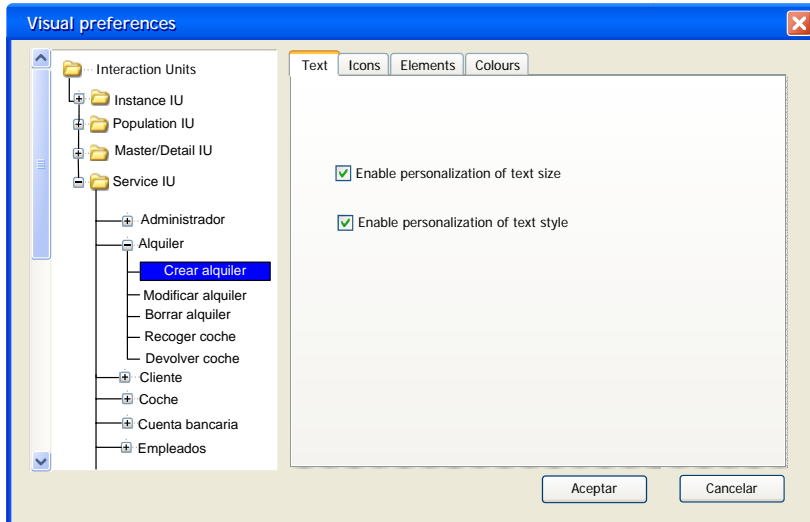


Figura 11.22 Modelo de Interacción Concreto para modelar WU_P1 (texto)

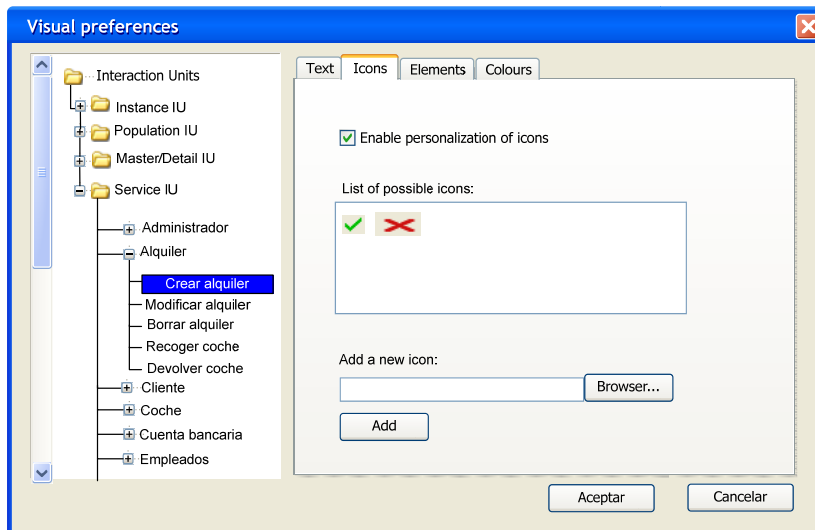


Figura 11.23 Modelo de Interacción Concreto para modelar WU_P1 (iconos)

La Figura 11.23 muestra el prototipo para habilitar o deshabilitar la personalización de los iconos de las Unidades de Interacción generadas. Además, en esta ventana, el analista debe cargar los iconos que el usuario podrá seleccionar en la interfaz generada según sus necesidades. La Figura 11.23 muestra todas las personalizaciones habilitadas.

La Figura 11.24 presenta el prototipo para modelar la habilitación de las preferencias relacionadas con elementos de la interfaz. En concreto, se puede habilitar o deshabilitar la personalización de los argumentos visibles y la modificación de los alias de los argumentos. La Figura 11.24 muestra todas las personalizaciones habilitadas.

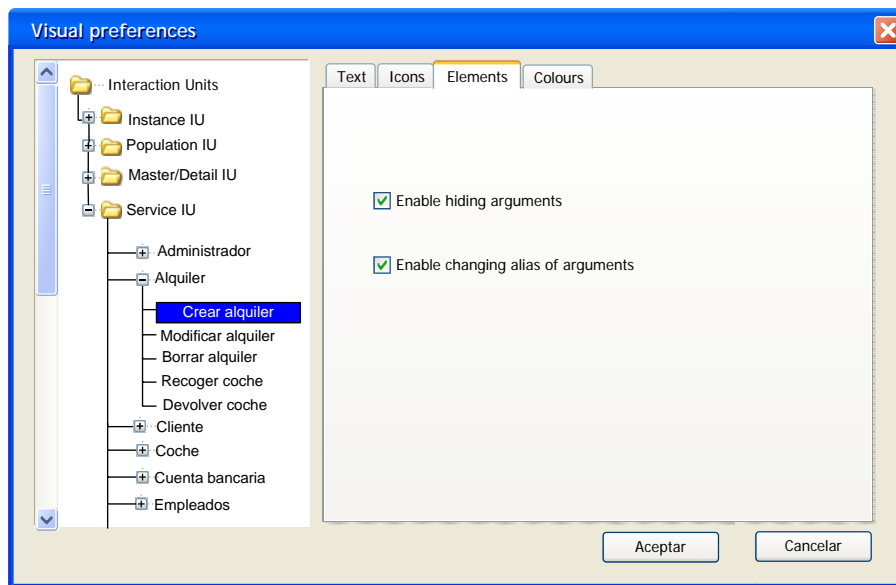


Figura 11.24 Modelo de Interacción Concreto para modelar WU_P1 (elementos)

Por último, la Figura 11.25 presenta el prototipo para modelar la habilitación de la personalización relacionada con los colores. Sólo existe la posibilidad de habilitar o deshabilitar la personalización de los colores del fondo de las Unidades de Interacción generadas. El ejemplo de la Figura 11.25 muestra esta personalización habilitada.

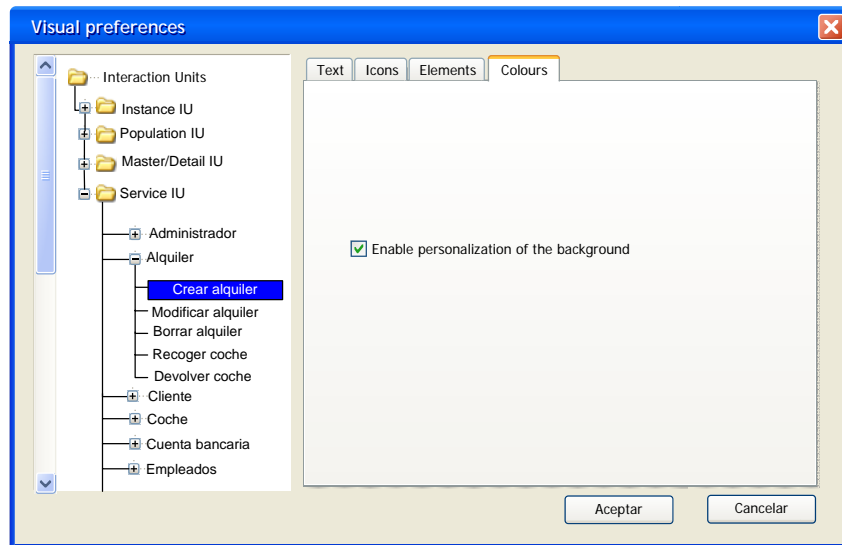


Figura 11.25 Modelo de Interacción Concreto para modelar WU_P1 (colores)

11.3.4 Modelado de WU_POS1

Esta sección explica cómo modelar la forma de uso *Distribución de elementos (WU_POS1)* en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan todas las Unidades de Interacción.

WU_POS1 incorpora un gran número de primitivas conceptuales para habilitar o deshabilitar la posibilidad de que el usuario modifique la posición de los elementos de las interfaces generadas. Estas primitivas se han agrupado en cuatro grupos distintos dependiendo del tipo de movimiento que habiliten: ordenación, alineación, tamaño, otros.

La Figura 11.26 presenta un prototipo de ventana para modelar WU_POS1. En la porción izquierda de la ventana se elige, mediante un menú en forma de árbol, la UI sobre la que se va a modificar la capacidad de personalizar la posición de alguno de sus elementos. En la porción derecha de la ventana se especifica para una UI específica, qué elementos podrá modificar su posición el usuario en la interfaz generada y cuáles no. En la Figura 11.26 se pueden apreciar los cuatro tipos distintos de movimientos de los que

puede disponer el usuario. Todos ellos están habilitados en el caso de ejemplo.

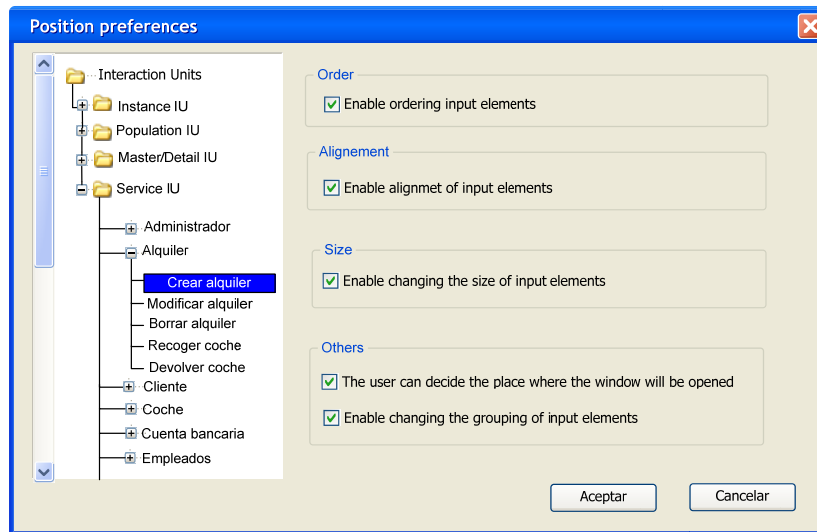


Figura 11.26 Modelo de Interacción Concreto para modelar WU_POS1

11.3.5 Modelado de WU_AO1

Esta sección explica cómo modelar la forma de uso *Cancelar durante la ejecución (WU_AO1)* en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan las UI de Servicio.

Los aspectos visuales a modelar para WU_AO1 se pueden dividir en dos grupos: generales y de formato. La Figura 11.27 muestra un prototipo de ventana para modelar las propiedades generales. En el menú en forma de árbol de la porción izquierda de la ventana, se selecciona la UI de Servicio sobre la cuál se va a modelar el aspecto visual para cancelar su ejecución. En este menú sólo aparecen UI de Servicio porque son las únicas UI que contienen servicios para ejecutar. En el ejemplo se ha seleccionado la UI de Servicio *Crear alquiler*. En la porción derecha de la ventana, el analista puede modelar para la UI de Servicio seleccionada, las propiedades visuales generales: posición del botón de cancelación; apariencia visual; teclas de acceso rápido. En el ejemplo de la Figura 11.27, el analista ha

elegido mostrar la opción de cancelar en una ventana emergente, mediante texto e icono, y la combinación de teclas de acceso rápido CTRL+A.

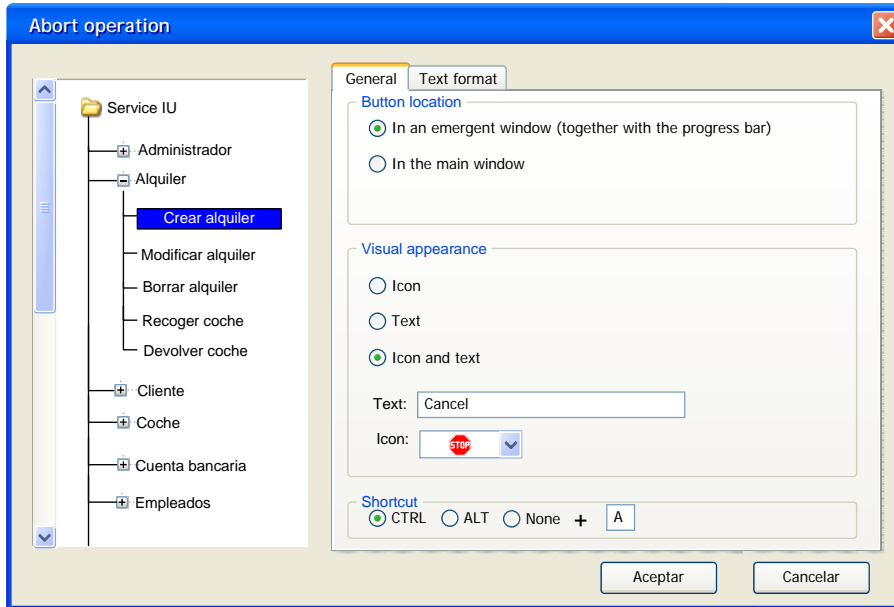


Figura 11.27 Modelo de Interacción Concreto para modelar WU_AO1 (1)

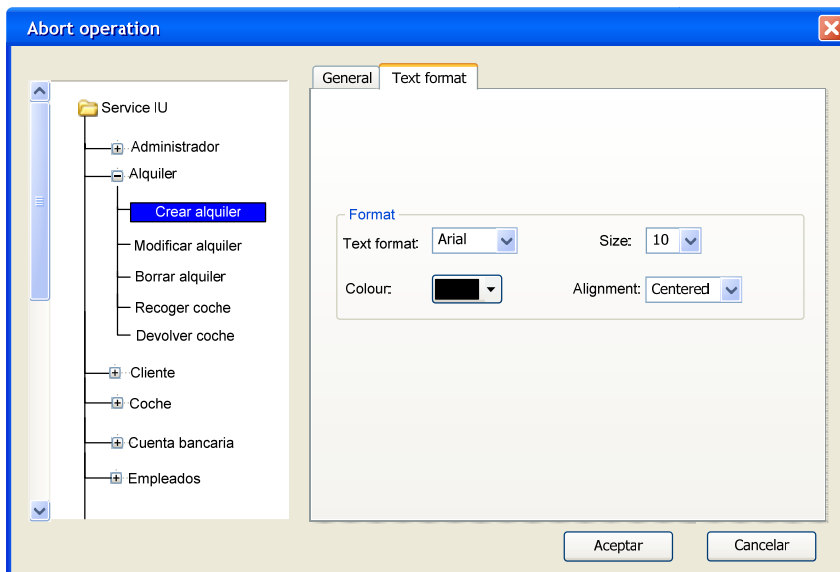


Figura 11.28 Modelo de Interacción Concreto para modelar WU_AO1 (2)

La Figura 11.28 muestra el prototipo para modelar el formato del texto del botón que realiza la cancelación del servicio para el servicio seleccionado. En este caso, el formato es de fuente Arial, tamaño 10, color negro y alineación centrada.

11.3.6 Modelado de WU_AO2

Esta sección explica cómo modelar la forma de uso *Salir de un escenario (WU_AO2)* en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan todas las Unidades de Interacción.

La Figura 11.29 muestra un prototipo de ventana sobre la que se modelaría WU_AO2 en el Modelo de Interacción Concreto. En la porción izquierda de la ventana se elige, en el menú en forma de árbol, la UI en la que se va a modelar la función de salir. Una vez seleccionada la UI, en la porción derecha de la ventana, se indica en qué posición de la UI se va a visualizar el botón de salir. En el ejemplo de la Figura 11.29 se muestra el lugar en el que visualizará el botón de *OK* y de *Cancel* para la UI de Servicio, *Crear alquiler*.

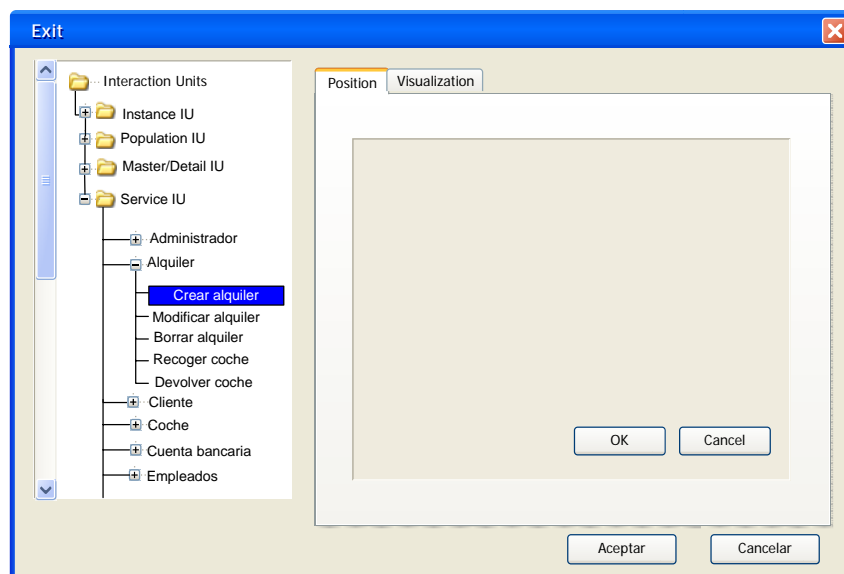


Figura 11.29 Modelo de Interacción Concreto para modelar WU_AO2 (1)

La Figura 11.30 muestra el prototipo para modelar detalles relacionados con la visualización de la funcionalidad de WU_AO2. Concretamente, se puede especificar el texto del botón que activa la salida del escenario, el icono, el formato del texto y las teclas de acceso rápido. En el ejemplo, se ha elegido como texto *Cancel*, un icono que acompañará al texto, formato Arial negro de tamaño 10 y centrado, teclas de acceso rápido mediante ALT+ F4.

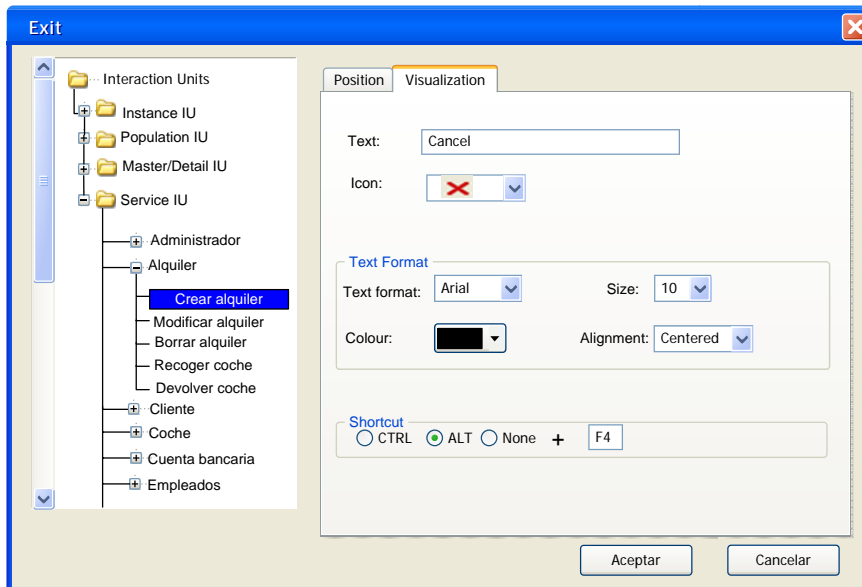


Figura 11.30 Modelo de Interacción Concreto para modelar WU_AO2 (2)

11.3.7 Modelado de WU_MH1

Esta sección explica cómo modelar la forma de uso *Ayuda dinámica* (*WU_MH1*) en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan todas las Unidades de Interacción.

La Figura 11.31 presenta un prototipo de una ventana desde la cual modelar WU_MH1 dentro del Modelo de Interacción Concreto. En la porción izquierda de la ventana, el analista elegiría la UI sobre la cual se va a modelar la visualización de la ayuda dinámica. En la porción derecha de la ventana se modelarían las propiedades visuales del mensaje de ayuda, es decir, se va a indicar si el mensaje se visualizará en un tooltip o dentro de la

ventana. En caso de que se muestre dentro de la ventana, el analista podrá especificar en qué posición de la ventana se mostrará la información. En el ejemplo de la Figura 11.31 se ha elegido que se visualizará la ayuda en la parte derecha de la interfaz. Dicha elección se define en la sección *Position of the help in the Window*.

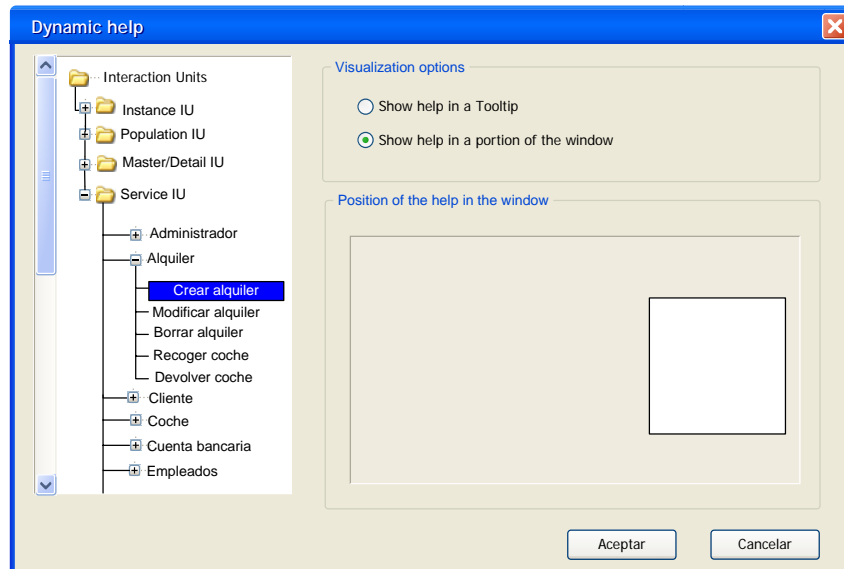


Figura 11.31 Modelo de Interacción Concreto para modelar WU_MH1

Esta forma de uso es la última que afecta al modelado de las Unidades de Interacción dentro del Modelo de Interacción Concreto. Una vez definida la visualización de las Unidades de Interacción, el siguiente paso es el de definir las propiedades para la visualización de los Patrones Elementales. A continuación se va a comentar cómo las formas de uso afectan a los Patrones Elementales en el Modelo de Interacción Concreto.

11.3.8 Modelado de WU_PF1

Esta sección explica cómo modelar la forma de uso *Mostrar progreso de la ejecución (WU_W1)* en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se visualizan los Patrones Elementales de acción en la interfaz generada.

La Figura 11.32 muestra el prototipo de una ventana para modelar WU_PF1 en el Modelo de Interacción Concreto. En la porción izquierda de la ventana, el analista elegiría sobre qué Patrón Elemental de acción va a modelar el aspecto visual de la barra de progreso. En el menú en forma de árbol, el analista sólo puede seleccionar aquellas acciones que representan la ejecución de una transacción y además se haya indicado en el Modelo de Interacción Abstracto que su ejecución implicaba el visualizar una barra de progreso (Figura 11.10). Una vez elegida la acción, en la porción de ventana derecha se especifican las propiedades de visualización de la barra. Estas propiedades se han dividido en dos grupos: gráficas y textuales. Concretamente, la Figura 11.32 muestra las propiedades gráficas. En esta ventana el analista podría decidir dónde ubicar la barra de progreso, si muestra el porcentaje y el tiempo restante, y el sentido de desplazamiento de la barra. En el ejemplo, se está modelando la barra de progreso para la transacción *Poner en venta* de la clase *Coche*. La barra se mostrará en la parte inferior derecha de la ventana principal y se desplazará de izquierda a derecha.

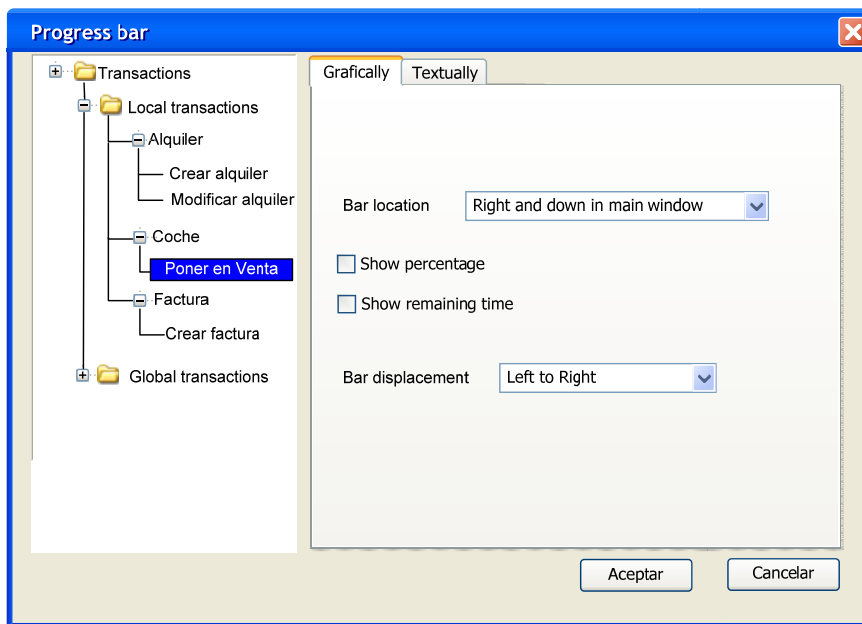


Figura 11.32 Modelo de Interacción Concreto para modelar WU_PF1 (1)

Por otro lado, la Figura 11.33 muestra las características de la barra de progreso en caso de que el analista quiera visualizar la barra de progreso en forma de una lista de texto que muestre los servicios que componen la transacción que ya se han ejecutado. Entre las propiedades que puede elegir el analista se encuentran: indicar dónde se va a ubicar la lista de servicios; si se van a listar los servicios ya completados y los restantes, si se van a listar sólo los completados, si se van a listar sólo los restantes; aspectos relacionados con el formato utilizado en el texto de la lista de servicios. En el ejemplo de la Figura 11.33, el analista ha elegido mostrar la lista de servicios ejecutados en una nueva ventana, listando todos los servicios que componen la transacción y resaltando el servicio que se está ejecutando en cada momento. Además, también se ha elegido un formato específico para el texto que liste los servicios.

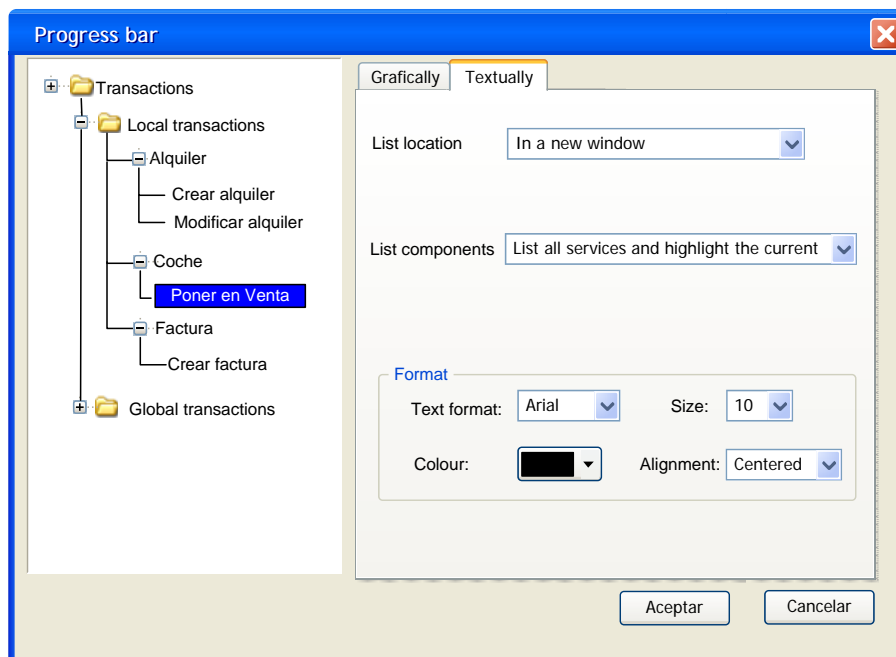


Figura 11.33 Modelo de Interacción Concreto para modelar WU_PF1 (2)

11.3.9 Modelado de WU_SSF1

Esta sección explica cómo modelar la forma de uso *Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)* en el Modelo de Interacción

Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan los Patrones Elementales de acción.

Son muchas las posibilidades para visualizar los mensajes de éxito y de fracaso, por lo que se han agrupado las primitivas de modelado en dos grupos: las relacionadas con el formato del mensaje y las relacionadas con la posición del mensaje en la interfaz. Cada uno de estos grupos da lugar a una pestaña dentro del prototipo de ventana que implementa el modelado de WU_SSF1. Además, habría una ventana para modelar la visualización de los mensajes de éxito y otra ventana distinta para modelar los mensajes de error.

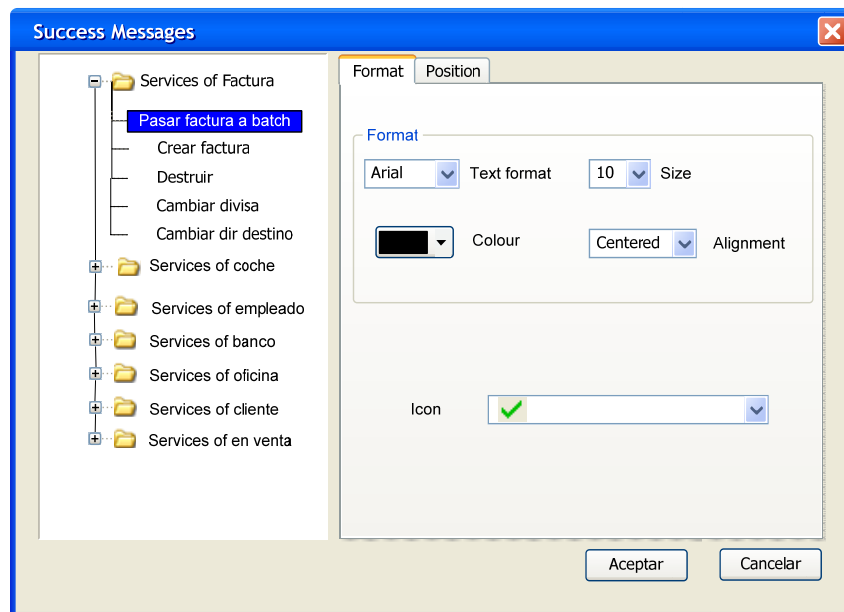


Figura 11.34 Modelo de Interacción Concreto para modelar WU_SSF1 (1)

La Figura 11.34 muestra la ventana desde la que se puede modelar el formato para el mecanismo WU_SSF1, en el caso de mostrar un mensaje de éxito. En la porción izquierda de la ventana se elige la acción sobre la que se va a modelar el formato de WU_SSF1. En la porción derecha de la ventana se muestran las distintas posibilidades de modelado del formato para WU_SSF1. Para el caso de ejemplo de la Figura 11.34, el analista ha

seleccionado para el servicio *Pasar factura a batch*, que pertenece a la clase del Modelo de Objetos *Factura*, las opciones de formato Arial de tamaño 10, color negro y alineación centrada, y un icono específico.

La Figura 11.35 muestra el prototipo para modelar la posición de la interfaz que muestra al usuario el mensaje derivado de WU_SSF1. En el menú en forma de árbol de la porción izquierda de la ventana, se elige la acción sobre la que se va a modelar el formato de WU_SSF1. En la porción derecha de la ventana se elige dónde se va a mostrar el mensaje de WU_SSF1. Las distintas posibilidades son: no mostrar el mensaje; mostrarlo en la ventana principal; mostrarlo en una ventana emergente. Además, cada opción tiene distintas primitivas para modelar cada una de las propiedades de WU_SSF1. En el caso de ejemplo de la Figura 11.35, se está modelando la visualización de WU_SSF1 para la acción *Pasar factura a batch*. El mensaje se mostrará en una ventana emergente, de tipo información y que será bloqueante.

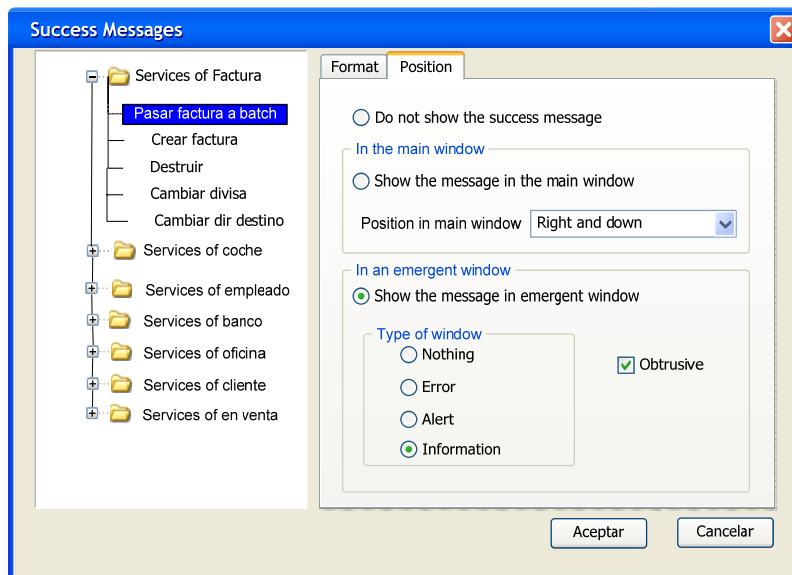


Figura 11.35 Modelo de Interacción Concreto para modelar WU_SSF1 (2)

Los prototipos de ventanas para los mensajes de error serían los mismos que los mostrados para los mensajes de éxito (Figura 11.34 y Figura 11.35).

11.3.10 Modelado de WU_SSF2

Esta sección explica cómo modelar la forma de uso *Mostrar el estado de la información (WU_SSF2)* en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifica cómo se representan los Patrones Elementales de navegación.

La Figura 11.36 muestra un prototipo de ventana desde la que se modela WU_SSF2. En la porción izquierda de la ventana, el analista puede elegir sobre qué navegación de qué clase de las definidas en el Modelo de Objetos va a modelar su visualización. Una vez elegida la navegación, en la porción derecha se pueden elegir las distintas alternativas de visualización: tamaño del botón de navegación; icono; formato del texto de la etiqueta del botón. En el ejemplo de la Figura 11.36 se ha elegido la navegación de la clase *Alquiler* hacia el listado de *Reservas*. El tamaño del botón de navegación será acorde al tamaño de su etiqueta, y se ha especificado su icono y su formato.

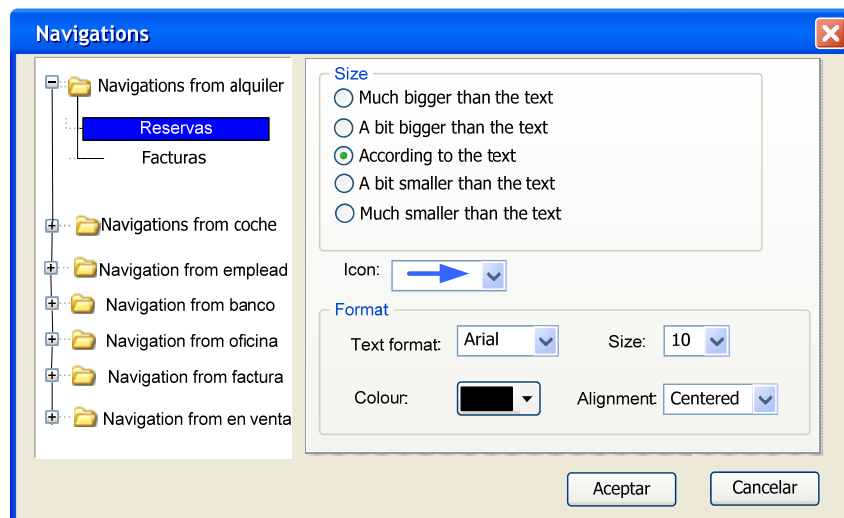


Figura 11.36 Modelo de Interacción Concreto para modelar WU_SSF2

11.3.11 Modelado de WU_W1

Esta sección explica cómo modelar la forma de uso *Mensaje de aviso (WU_W1)* en el Modelo de Interacción Concreto de OO-Method. Esta forma

de uso se modela dentro de la sección donde se especifican los Patrones Auxiliares de OO-Method. En este caso, para cada uno de los mensajes de aviso definidos en el Modelo de Objetos, el analista podría especificar cómo se visualizará dicho mensaje al usuario.

La Figura 11.37 muestra un prototipo de ventana desde el cual modelar WU_W1 dentro del Modelo de Interacción Concreto. En la porción izquierda de la ventana, aparecen los distintos mensajes de aviso que se han definido en el Modelo de Objetos en forma de árbol. Los mensajes van agrupados por la clase en la cual se definieron. En la porción derecha de la ventana se muestran las distintas primitivas con las que modelar las propiedades configurables de WU_W1: si el mensaje será o no bloqueante; el tipo de ventana en el cual se mostrará el mensaje; el formato para el mensaje. En el ejemplo de la Figura 11.37 se puede apreciar que se ha elegido un mensaje de aviso definido sobre el servicio *Crear alquiler* que pertenece a la clase *Alquiler*. El analista ha elegido como propiedades que sea un mensaje bloqueante, de tipo alerta y con un formato específico.

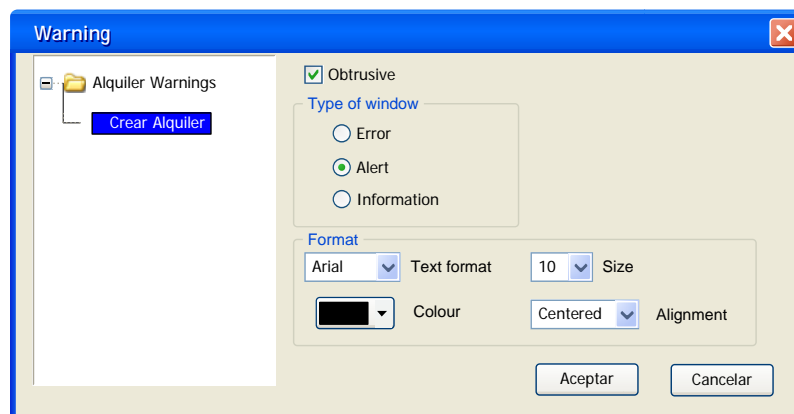


Figura 11.37 Modelo de Interacción Concreto para modelar WU_W1

11.3.12 Modelado de WU_F1

Esta sección explica cómo modelar la forma de uso *Definición de favoritos* (WU_F1) en el Modelo de Interacción Concreto de OO-Method. Esta forma de uso se modela dentro de la sección donde se especifican los Patrones

Auxiliares de OO-Method, más concretamente cuando se modelan los aspectos generales del sistema

Es el propio usuario el que define la lista de favoritos, por lo que en este modelo el analista sólo habilita la capacidad del usuario para gestionar la lista de favoritos. La Figura 11.38 muestra un prototipo de ventana donde el analista podría decidir si habilita o no la gestión de la lista de favoritos. Esta ventana debe estar accesible desde las herramientas de configuración de los Patrones Auxiliares.

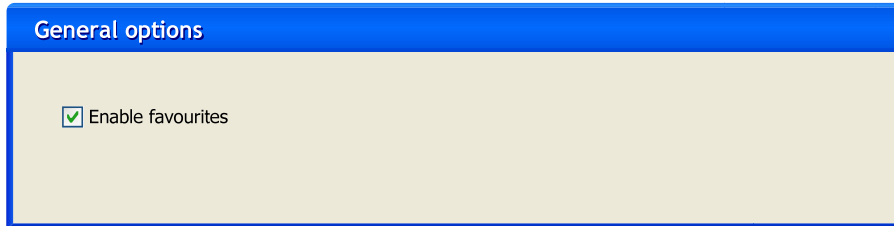


Figura 11.38 Modelo de Interacción Concreto para modelar WU_F1

11.4 Conclusiones de la viabilidad

En este capítulo se ilustra cómo incorporar las propiedades configurables que se han extraído de las formas de uso en capítulos anteriores. Este ejercicio sirve para demostrar la viabilidad de incorporar las propiedades configurables a una TTM específica, como es OO-Method. Este capítulo se puede utilizar como guía para que el analista sepa cómo trabajar con las formas de uso dentro de OO-Method. Además, este estudio de viabilidad puede servir a los diseñadores de otras TTM para deducir cómo se modela en sus aproximaciones cada una de las propiedades configurables identificadas a lo largo de la tesis. Lo único que cambia son los modelos conceptuales sobre los que hay que modelar las propiedades configurables, pero las abstracciones para representar las propiedades serán similares.

Es importante resaltar que la explicación sobre cómo modelar las distintas propiedades configurables no incluye la definición de guías para obtener la máxima usabilidad en las aplicaciones generadas. Es decir, la existencia de las primitivas que representan las formas de uso no garantiza que el

sistema sea usable, sino que la usabilidad del sistema depende de las decisiones que tome el analista en la fase de modelado. Como trabajo futuro se propondrá una serie de guías de buenas prácticas explicando las combinaciones que obtienen mejor valor para la usabilidad de los sistemas. Estas guías deben estar basadas en una serie de experimentos con usuarios que corroboren las mejores combinaciones de las primitivas para conseguir un buen nivel de usabilidad.

Capítulo 12

Evaluación Experimental

Tal y como se ha comentado en capítulos anteriores, la incorporación de los mecanismos de usabilidad (divididos en formas de uso) con el método MIMAT implica cambios en los modelos conceptuales y en el compilador de modelos de la TTM sobre la que se hace la incorporación. Esta sección evalúa si la aplicación del método MIMAT mejora la usabilidad de las aplicaciones generadas mediante la TTM. Esta evaluación justifica el esfuerzo que supone la modificación del modelo conceptual y del compilador de modelos de la TTM.

Como ejemplo, se ha utilizado una aplicación Web desarrollada con la herramienta que implementa OO-Method: OlivaNOVA. Con OlivaNOVA se ha modelado y generado el código de una aplicación en ASP .NET que implementa el sistema de alquiler de coches que ha servido de caso de ilustración durante toda la tesis (Figura 4.5). A partir de la aplicación generada se han hecho dos grupos de usuarios:

- **Usuarios que interactúan con una aplicación que no incorpora formas de uso:** Este grupo de usuarios interactúa con la aplicación Web tal cual la genera el compilador de modelos de OlivaNOVA. Esta aplicación no incluye ninguna forma de uso, tampoco las actualmente soportadas por OO-Method.
- **Usuarios que interactúan con una aplicación que incorpora formas de uso:** Este grupo de usuarios interactúa con la aplicación Web de alquiler de coches a la cual se han añadido varias formas de uso. Algunas de estas formas de uso ya están soportadas por OO-Method, pero otras se han incluido de manera manual. La incorporación manual se ha hecho modificando el código generado

por OlivaNOVA, ya que su compilador de modelos aún no soporta su implementación.

El estudio sobre las mejoras en usabilidad tras aplicar el método MIMAT a OO-Method se hace en base a las opiniones de los usuarios de cada grupo.

En las siguientes secciones se expone qué formas de uso se han seleccionado para formar parte del experimento, cómo se ha diseñado el experimento, la relación entre formas de uso y atributos de usabilidad utilizada para medir la usabilidad, los instrumentos utilizados en la evaluación, las dos aplicaciones Web utilizadas en el experimento (una con formas de uso y otra sin ellas) y el análisis de los datos extraídos.

12.1 Formas de uso utilizadas en el experimento

El principal objetivo de evaluar la mejora de la usabilidad en las aplicaciones generadas es el de aportar evidencias a los diseñadores de herramientas MDD de la necesidad de incorporar mecanismos de usabilidad dentro del proceso de desarrollo. Por tanto, no es crítico que el experimento se haga con todas las formas de uso que se han definido en la tesis. Con un subconjunto es suficiente para conseguir nuestro propósito.

La mayoría de las formas de uso elegidas para la aplicación Web que incorpora formas de uso no están soportadas actualmente por OlivaNOVA. A continuación se listan las formas de uso agrupadas por mecanismos de usabilidad que se han utilizado en la aplicación Web de alquiler de coches:

- **System Status Feedback:**
 - Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1): El sistema muestra para cada ejecución de un servicio, si ha finalizado o no con éxito
 - Mostrar el estado de las acciones (WU_SSF3): Esta forma de uso hace que aquellas acciones que incumplen alguna precondición se inhabiliten. De esta forma se evitan posibles errores por parte de los usuarios.

- **Warning:**
 - Mensaje de aviso (WU_W1): Esta forma de uso se ha aplicado al servicio *Reservar alquiler de coche*. El sistema avisa al usuario si las fechas de alquiler tienen una duración superior a un mes.

- **Structured Text Entry:**
 - Especificar el tipo de visualización del campo de entrada (WU_STE1): Al aplicar esta forma de uso, el analista puede adaptar el tipo de argumento de entrada según lo más adecuado para el usuario. WU_STE1 está en parte soportada por OlivaNOVA. Aquellos campos de entrada no soportados se han implementado manualmente.
 - Definición de máscaras (WU_STE2): Esta forma de uso ya está actualmente soportada por OlivaNOVA y por tanto no ha sido necesario hacer modificaciones manuales en el código generado. WU_STE2 se ha aplicado en varios campos de entrada de datos que requieren un formato específico.
 - Valores por defecto (WU_STE3): Esta forma de uso también está actualmente soportada por OlivaNOVA. Al aplicar WU_STE3, se muestra al usuario valores por defecto en los campos donde el valor a introducir se pueda predecir.

- **Multilevel help:**
 - Ayuda dinámica (WU_MH1): Esta forma de uso hace que el sistema muestre texto de ayuda en varias interfaces del sistema de manera dinámica, conforme a las necesidades del usuario. Es decir, dependiendo de las acciones que haga el usuario, se muestran o no mensajes de ayuda.

De todas estas formas de uso, las que ya están soportadas por OlivaNOVA son: *Definición de máscaras (WU_STE2)*, *Valores por defecto (WU_STE3)* y *Especificar el tipo de visualización del campo de entrada (WU_STE1)*. WU_STE1 está sólo soportada en parte y hubo que añadir código manual para darle soporte.

El resto de las formas de uso definidas en la tesis no se han incorporado por varias razones. A continuación se justifica para cada forma de uso, porque no se ha utilizado en la evaluación.

- **System Status Feedback:**
 - Mostrar el estado de la información (WU_SSF2): Esta forma de uso se usa para mostrar información almacenada que puede ser útil conocer antes de ejecutar un servicio. Su implementación en las aplicaciones generadas por OlivaNOVA es muy costosa. Además, la información almacenada en las aplicaciones generadas con OlivaNOVA se puede consultar desde cualquier interfaz por medio de navegaciones.
- **Interaction Feedback:**
 - Informar que la interacción está siendo atendida (WU_IF1): Esta forma de uso no es crítica para las aplicaciones Web porque existe la barra de progreso del navegador Web para indicar que la petición de ejecución es atendida.
- **Progress Feedback:**
 - Mostrar progreso de la ejecución (WU_PF1): Los servicios definidos en la aplicación de alquiler de coches son sencillos y requieren poco tiempo para su ejecución. Por tanto, no es necesaria la aplicación de esta forma de uso.
- **Step by Step:**
 - Definir un asistente (WU_SBS1): No existen tareas lo suficientemente complejas en la aplicación de alquiler de coches como para añadir esta forma de uso.
- **Preferences:**
 - Preferencias en el aspecto visual (WU_P1): El sistema de alquiler de coches no está pensado para un amplio número de sujetos, sino sólo para los trabajadores de la empresa. Por lo tanto, el que el número de usuarios no sea elevado resta importancia a la incorporación de esta forma de uso.

- Preferencias en el idioma (WU_P2): Todos los sujetos que han participado en la prueba son españoles. Por tanto, carece de sentido el proporcionar la aplicación Web en varios idiomas.
- **Personal Object Space:**
 - Distribución de elementos (WU_POS1): La razón por la que no se ha incorporado esta forma de uso es la misma que por la que no se ha incorporado WU_P1, el número de usuarios que utilizará la aplicación es reducido.
- **Favourites:**
 - Definición de favoritos (WU_F1): Esta forma de uso tiene sentido aplicarla en sistemas en los que el usuario utiliza ciertos servicios reiteradamente. En el caso del sistema de alquiler de coches, los sujetos interactúan con el sistema durante un breve periodo de tiempo, por lo que no se puede apreciar la utilidad de definir favoritos.
- **Global undo:**
 - Deshacer cambios (WU_GU1): Esta forma de uso es una de las más complejas de incorporar en cualquier sistema, tal y como establece Juristo en un estudio [Juristo, 2007, a].
 - Rehacer cambios (WU_GU2): No se puede incorporar si no se ha incorporado previamente la forma de uso WU_GU1.
- **Abort operation:**
 - Cancelar durante la ejecución (WU_AO1): Esta forma de uso está especialmente pensada para aquellos servicios que requieren mucho tiempo para su ejecución. No hay ningún servicio en la aplicación de alquiler de coches que requiera más de un segundo.
 - Salir de un escenario (WU_AO2): La aplicación Web de alquiler de coches no requiere navegaciones en profundidad entre sus contextos. Por tanto, el uso de esta forma de uso no es crítica para el caso de ilustración.
- **Multilevel help:**

- Ayuda estática (WU_MH2): La funcionalidad del alquiler de coches es lo suficientemente sencilla e intuitiva como para necesitar ayuda estática a parte de la ayuda dinámica ya incorporada (WU_MH1).

La Tabla 12.1 muestra un resumen de las formas de uso incorporadas.

Mecanismo	Formas de uso	Incorporada	Contemplada por OlivaNOVA
System Status Feedback	Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)	Sí	No
	Mostrar el estado de la información (WU_SSF2)	No	No
	Mostrar el estado de las acciones (WU_SSF3)	Sí	No
Interaction Feedback	Informar que la interacción está siendo atendida (WU_IF1)	No	No
Progress Feedback	Mostrar progreso de la ejecución (WU_PF1)	No	No
Warning	Mensaje de aviso (WU_W1)	Sí	No
Step by Step	Definir un asistente (WU_SBS1)	No	No
Structured Text Entry	Especificar el tipo de visualización del campo de entrada (WU_STE1)	Sí	No
	Definición de máscaras (WU_STE2)	Sí	Sí

Mecanismo	Formas de uso	Incorporada	Contemplada por OlivaNOVA
	Valores por defecto (WU_STE3)	Sí	Sí
Preferences	Preferencias en el aspecto visual (WU_P1)	No	No
	Preferencias en el idioma (WU_P2)	No	Sí
Personal Object Space	Distribución de elementos (WU_POS1)	No	No
Favourites	Definición de favoritos (WU_F1)	No	No
Global Undo	Deshacer cambios (WU_GU1)	No	No
	Rehacer cambios (WU_GU2)	No	No
Abort Operation	Cancelar durante la ejecución (WU_AO1)	No	No
	Salir de un escenario (WU_AO2)	No	No
Multilevel Help	Ayuda dinámica (WU_MH1)	Sí	No
	Ayuda estática (WU_MH2)	No	No

Tabla 12.1 Resumen de las formas de uso incorporadas en el experimento

12.2 Diseño del experimento

El experimento se ha diseñado siguiendo a los trabajos de Basili [Basili, 88] y de Wohlin [Wohlin, 99]. En las siguientes subsecciones se discute cada elemento experimental.

12.2.1 Objetivos

- **Objetivo general:** demostrar que la aplicación del método MIMAT a OO-Method mejora la usabilidad de las aplicaciones generadas.
- **Objetivos específicos:**

La Tabla 12.2 muestra los objetivos específicos del experimento siguiendo el método Goal/Question/Metric de Basili.

Analizar	Mecanismos de usabilidad y sus respectivas formas de uso
Con el propósito de	Evaluar cómo se modifica la usabilidad percibida por el usuario tras incorporar las formas de uso
Con respecto a	Satisfacción del usuario y eficiencia
Desde el punto de vista	Del investigador
En el contexto de	El estudio se realiza en un ambiente académico con sujetos que tienen experiencia previa en el uso de aplicaciones Web, quienes interactúan con una aplicación Web generada por OlivaNOVA. Un grupo interactúa con la aplicación usando formas de uso y el otro grupo interactúa con la misma aplicación pero sin usar formas de uso.

Tabla 12.2 Objetivos específicos del experimento

12.2.2 Preguntas y métricas

- **Preguntas:**
 - **P1:** ¿La satisfacción de los usuarios que interactúan con la aplicación Web que incluye formas de uso es mejor que la satisfacción de los usuarios que interactúan con aplicaciones Web que no incluye formas de uso?
 - **P2:** ¿La eficiencia de los usuarios que interactúan con la aplicación Web que incluye formas de uso es mejor que la eficiencia de los usuarios que interactúan con aplicaciones Web que no incluye formas de uso?
- **Métricas:**
 - **M1:** Se ha dividido cada forma de uso en los atributos de usabilidad con los que está relacionada. Como atributos de usabilidad se han utilizado los definidos en la ISO/IEC 9126-1 [ISO/IEC 9126-1, 2001], los trabajos de Bastien [Bastien, 1993] y los de Abrahao [Abrahao, 2006] [Abrahao, 2005] [España, 2006]. Para cada atributo de cada forma de uso se ha redactado una pregunta que capture la opinión del usuario sobre el valor de ese atributo en la aplicación Web. Las preguntas tienen respuestas en una escala ordinal likert de 5 puntos. La satisfacción del usuario se obtiene mediante sus respuestas a dichas preguntas. En la siguiente sección se explica el proceso en detalle.
 - **M2:** Número de minutos que el usuario tarda en realizar cada una de las tareas que componen el experimento. Se ha cronometrado el tiempo que el usuario tarda en completar cada una de las tareas de manera independiente al resto.
- **Selección de sujetos:** La selección ha sido por conveniencia, es decir, se han elegido los sujetos más cercanos y los más convenientes para la prueba. Estos sujetos tienen que cumplir una serie de características: deben ser personas con experiencia previa en el uso de aplicaciones Web pero los conocimientos detallados de informática y de modelado no son un requisito; el número de sujetos

estudiados oscila entre 40 y 70; el rango de edad de estos sujetos está entre 15 y 55 años.

12.2.3 Planificación detallada

- **Selección de sujetos:** Las características de todos los usuarios son parecidas. Personas con experiencia previa en el uso de aplicaciones Web pero que no tienen porqué tener conocimientos detallados de informática ni de modelado.
- **Selección de variables:**
 - **Variables independientes (factores):**
 - Utilización de las formas de uso en el sistema
 - Experiencia del usuario en aplicaciones generadas por OlivaNOVA
 - **Variables dependientes:**
 - Grado de satisfacción del usuario
 - Tiempo en realizar las tareas
 - **Parámetros:**
 - El dominio de la aplicación utilizada en el caso de ilustración
 - La experiencia previa de los usuarios en el uso de aplicaciones Web
- **Hipótesis nula:**
 - **H₁₀:** La satisfacción de los usuarios que usan aplicaciones Web con formas de uso es igual a la satisfacción de los usuarios que usan aplicaciones Web sin formas de uso
 - **H₂₀:** La eficiencia de los usuarios que usan aplicaciones Web con formas de uso es igual a la eficiencia de los usuarios que usan aplicaciones Web sin formas de uso
- **Hipótesis alternativas:**

- **H1₁**: La satisfacción de los usuarios que usan aplicaciones Web con formas de uso es mayor que la satisfacción de los usuarios que usan aplicaciones Web sin formas de uso
- **H2₁**: La eficiencia de los usuarios que usan aplicaciones Web con formas de uso es mayor que la eficiencia de los usuarios que usan aplicaciones Web sin formas de uso
- **Instrumentación**: el experimento ha contado con los siguientes instrumentos:
 - **Cuestionario demográfico**: Se ha diseñado un cuestionario para capturar información personal de los usuarios: género, edad, profesión, experiencia previa en el uso de aplicaciones Web, experiencia previa en el uso de aplicaciones generadas por OlivaNOVA.
 - **Tareas**: Se han definido 4 tareas que deben realizar todos los usuarios. Cada una de estas tareas tienen el objetivo de estudiar la mejora en usabilidad que aportaría un conjunto de formas de uso. Las tareas son las mismas para los dos grupos de sujetos (los que interactúan con el sistema sin formas de uso y con ellas). Estas tareas aseguran que los usuarios interactúan de la misma manera con el sistema. Se anota la hora de inicio y de fin en cada una de las tareas. De esta manera se registra el tiempo que tarda cada usuario en realizar cada una de las tareas. Este tiempo se calcula de manera automática en el sistema desde el que se realiza la experimentación.
 - **Cuestionario para capturar la usabilidad percibida por el usuario**: Se ha usado un cuestionario con preguntas cerradas de escala Likert de 5 puntos para que los usuarios expresen el grado de usabilidad del sistema. Tras la finalización de cada tarea, se muestran una serie de preguntas relacionadas con las formas de uso tratadas en la tarea. Las preguntas van orientadas a capturar la valoración del usuario de los atributos relacionados con las formas de uso que son objeto de estudio en cada tarea. Al finalizar las 4 tareas y sus correspondientes preguntas, se han

preguntado cuestiones para recoger la evaluación de usabilidad de la aplicación general.

Todos estos instrumentos están soportados electrónicamente por una Web. De esta manera se favorece la difusión de la evaluación a cualquiera que tenga acceso a Internet. Los cuestionarios y las tareas se pueden consultar en <http://hci.dsic.upv.es/TareasEvaluacion>.

12.2.4 Diseño del experimento

En el experimento hay dos factores a considerar sobre una aplicación Web. Estos factores son la *Utilización de las formas de uso en el sistema* y la *experiencia del usuario en aplicaciones generadas por OlivaNOVA*. A continuación se explican estos factores y sus tratamientos:

- **Utilización de las formas de uso:** Este factor compara la satisfacción de usuarios que interactúan con la aplicación Web que incorpora formas de uso y usuarios que interactúan con la misma Web pero sin incorporar formas de uso. Por lo tanto, este factor tiene dos tratamientos. Un tratamiento es la aplicación con formas de uso y el otro tratamiento es la misma aplicación pero sin las formas de uso.
- **Experiencia del usuario en aplicaciones generadas por OlivaNOVA:** Este factor se usa para diferenciar los usuarios que alguna vez han interactuado con aplicaciones generadas por OlivaNOVA y los que interactúan por primera vez en el experimento. Por lo tanto, este factor tiene dos tratamientos: los usuarios con experiencia en aplicaciones generadas con OlivaNOVA y los que no la tienen.

El diseño del experimento consiste en combinar los diferentes valores que pueden tener los dos factores estudiados. Para este fin se ha diseñado un test factorial 2x2. Aplicando este test, resultan cuatro grupos de estudio:

- Sujetos con experiencia que interactúan con la aplicación que utiliza formas de uso.

- Sujetos sin experiencia que interactúan con la aplicación que utiliza formas de uso.
- Sujetos con experiencia que interactúan con la aplicación que no utiliza formas de uso.
- Sujetos sin experiencia que interactúan con la aplicación que no utiliza formas de uso.

La asignación de sujetos que interactúan con la aplicación Web con formas de uso o sin ellas se realiza al azar, pero siempre asegurando que ambos grupos están balanceados tanto para los sujetos con experiencia como para los que no tienen experiencia. En la Tabla 12.3 se muestra el número de sujetos de cada grupo que ha participado en el experimento.

		Utilización de formas de uso		
		Números de sujetos que usan la aplicación Web	Usando mecanismos	Sin usar mecanismos
Factor: Experiencia en aplicaciones generadas por OlivaNOVA	Usuario con experiencia	11	X	
		11		X
	Usuario sin experiencia	22	X	
		22		X

Tabla 12.3 Diseño factorial 2x2 del experimento

12.2.5 Proceso del experimento

La Figura 12.1 muestra un esquema gráfico con el proceso del experimento. El proceso empieza con un cuestionario demográfico. Una vez rellenado

este cuestionario, ya se sabe si el usuario tiene o no experiencia previa en el uso de aplicaciones desarrolladas con OlivaNOVA, y dependiendo de su condición se asigna a uno u otro grupo. De manera totalmente oculta para el usuario, se decide si éste interactuará con la aplicación que utiliza formas de uso o la que no las utiliza. Por ejemplo, si el sujeto pertenece al grupo de expertos y la última interacción de este grupo se hizo con la aplicación que utiliza formas de uso, la siguiente interacción es con la aplicación que no utiliza formas de uso. Así se asegura que los dos grupos de usuarios tengan interacciones balanceadas con las dos aplicaciones.

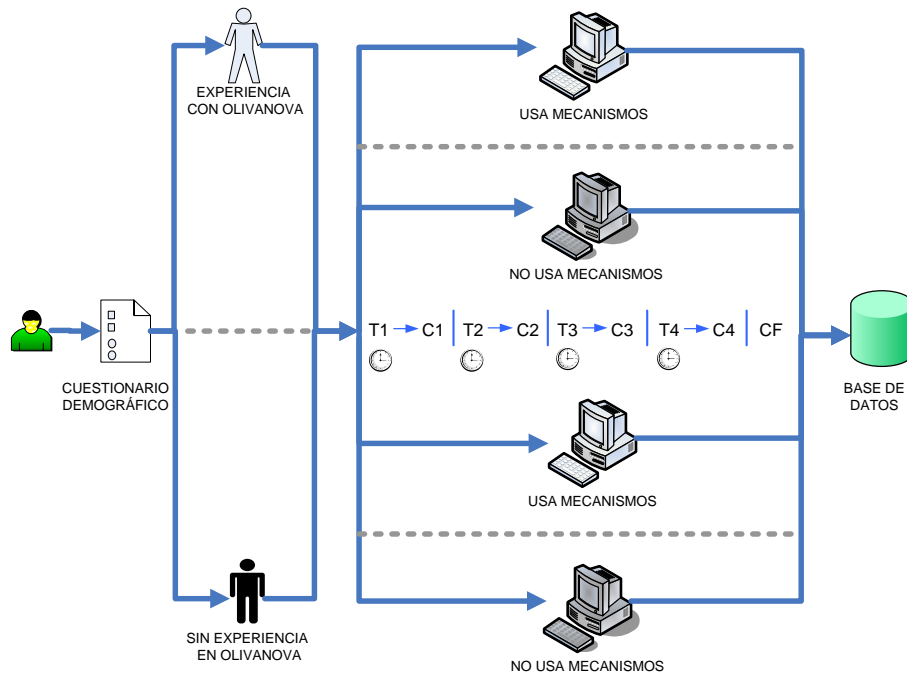


Figura 12.1 Proceso del experimento

Una vez decidida la aplicación, el siguiente paso es la ejecución de tareas. Hay un total de 4 tareas. Se cronometra de forma oculta para el usuario el tiempo que éste tarda en completar cada una de las 4 tareas. Al finalizar cada tarea, el usuario debe rellenar un cuestionario de escala likert de 5 puntos (C1, C2, C3, C4, C5 en Figura 12.1) donde se recoge la opinión de usabilidad de cada una de las formas de uso que se trataban en la tarea. Hay una pregunta por cada atributo de usabilidad relacionado con las formas de uso tratadas.

Por último, una vez rellenados los cuestionarios de las 4 tareas, hay un último cuestionario (*CF* en Figura 12.1) en el que se recoge la opinión del usuario sobre la usabilidad en general de toda la aplicación Web. Es importante resaltar que todo este proceso se realiza a distancia a través de una página Web que implementa las tareas y los cuestionarios. La información sobre el tiempo invertido en cada tarea y los resultados de los cuestionarios se almacenan en una base de datos.

12.2.6 Amenazas a la validez

- **Validez de conclusión:**
 - **Sujetos de heterogeneidad aleatoria:** Esta amenaza aparece si dentro del grupo de usuarios estudiados hay unos que tienen más experiencia en el uso de aplicaciones Web que otros. En la evaluación realizada, la mayoría de sujetos tienen amplia experiencia en el uso de aplicaciones Web. Además, se ha diferenciado entre usuarios con experiencia en el uso de aplicaciones desarrolladas por OlivaNOVA y usuarios sin experiencia.

- **Validez interna:**
 - **Maduración:** Esta amenaza contempla la posibilidad de que los usuarios reaccionen de manera distinta conforme transcurre el tiempo del experimento, bien por cansancio o por aburrimiento. Esta amenaza se ha intentado solucionar haciendo una evaluación corta. El tiempo estimado de toda la prueba era de 15 minutos.
 - **Instrumentación:** A pesar de que tanto tareas como cuestionarios son los mismos para todos los usuarios, éstos pueden sufrir distintas interpretaciones por parte del usuario que los lea. Para evitar esta amenaza, se hizo un pretest con tres usuarios con el fin de detectar las tareas y preguntas de los cuestionarios que eran ambiguas o difíciles de entender. Algunas tareas y preguntas se describieron de forma más precisas tras el pretest.

- **Validez de construcción:**
 - **Suposición de la hipótesis:** Esta amenaza contempla la posibilidad de que el usuario suponga el propósito y el resultado esperado del experimento y base su comportamiento en esas suposiciones. Esta amenaza se ha minimizado ocultando el objetivo del experimento.

- **Validez externa:**
 - **Interacción de la selección y los tratamientos:** Este es el efecto de tener una población que no sea representativa de la población a la que queremos generalizar. Esta amenaza se controla balanceando el número de usuarios con experiencia en aplicaciones generadas por OlivaNOVA y los que no la tienen. Además, también existe una amenaza a la hora de generalizar los resultados para otras aplicaciones Web de un dominio distinto (no es objetivo de la tesis). Esta amenaza desaparecería si el experimento se duplicara en aplicaciones Web de distintos dominios.

12.3 Relación entre formas de uso y atributos de usabilidad

Como se ha comentado en la sección anterior, cada una de las formas de uso afecta a uno o varios atributos de usabilidad. Es decir, la incorporación al sistema de una forma de uso mejora o empeora varios atributos de usabilidad. Esta sección identifica los atributos de usabilidad que están relacionados con cada una de las formas de uso utilizadas en el experimento.

Es importante resaltar que los atributos de usabilidad no tienen porqué estar relacionados con una única forma de uso de manera exclusiva. Los atributos de usabilidad son lo suficientemente abstractos como para abarcar el objetivo de varias formas de uso.

La relación entre atributos de usabilidad y formas de uso se utiliza para dirigir el estudio empírico hacia aquellos atributos que estén relacionados con las formas de uso que son estudiadas en el experimento. En base a estos atributos se han elaborado los cuestionarios que capturan la satisfacción por parte del usuario. La razón por el que se van a utilizar los atributos de usabilidad es porque según la ISO/IEC 9126-1 [ISO/IEC 9126-1, 2001], todos los atributos de usabilidad son entes medibles del concepto abstracto que es la usabilidad.

Los atributos de usabilidad que se han utilizado son los descritos en la ISO/IEC 9126-1 [ISO/IEC 9126-1, 2001] y el Modelo de Usabilidad propuesto por Abrahao [Abrahao, 2006] [Abrahao, 2005] [España, 2006]. Además, también se ha estudiado la relación entre las formas de uso y los criterios ergonómicos de Bastien y Scapin [Bastien, 1993]. Estos criterios ergonómicos son ampliamente utilizados dentro de la comunidad IPO para evaluar la usabilidad de sistemas. A continuación se presenta la lista de atributos y criterios ergonómicos de cada forma de uso agrupados por mecanismo de usabilidad.

12.3.1 Atributos de System Status Feedback

De este mecanismo de usabilidad se han utilizado en el experimento dos formas de uso distintas: *Informar del éxito o fracaso en la ejecución del servicio* y *Mostrar el estado de las acciones*. La primera de las formas de uso tiene como objetivo informar al usuario de si los servicios se han ejecutado o no correctamente. La Tabla 12.4 muestra los atributos de usabilidad y criterios ergonómicos que tienen este mismo objetivo.

Atributo Modelo Usabilidad Abrahao	Atributo de la ISO/IEC 9126-1	Criterio Ergonómico Bastien y Scapin
Realimentación informativa	Compleitud de la información	Realimentación inmediata
Calidad de los mensajes	Claridad del mensaje	Calidad de los mensajes de error

Tabla 12.4 Relación entre *Informar del éxito o fracaso en la ejecución del servicio*, atributos y criterios ergonómicos

En cuanto a la forma de uso *Mostrar el estado de las acciones*, tiene como objetivo evitar errores mostrando el estado de las acciones que puede lanzar el usuario. La Tabla 12.5 muestra los atributos y los criterios ergonómicos que tienen el mismo objetivo que dicha forma de uso.

Atributo Modelo Usabilidad Abrahao	Atributo de la ISO/IEC 9126-1	Criterio Ergonómico Bastien y Scapin
Realimentación informativa	Compleitud de la información	Realimentación inmediata
Calidad de los mensajes	Claridad del mensaje	Calidad de los mensajes de error
Prevención de errores	Validación de elementos de entrada	Protección a errores

Tabla 12.5 Relación entre *Mostrar el estado de las acciones*, atributos y criterios ergonómicos

12.3.2 Atributos de Warning

En el caso de ilustración del experimento se ha añadido la única forma de uso del mecanismo de usabilidad *Warning*, llamada *Mensaje de aviso*. La Tabla 12.6 muestra los atributos de usabilidad y criterios ergonómicos que tienen este mismo objetivo.

Atributo Modelo Usabilidad Abrahao	Atributo de la ISO/IEC 9126-1	Criterio Ergonómico Bastien y Scapin
Realimentación informativa	Compleitud de la información	Realimentación inmediata
Calidad de los mensajes	Claridad del mensaje	Calidad de los mensajes de error
Prevención de errores	Validación de elementos de entrada	Protección a errores

Tabla 12.6 Relación entre *Mensaje de aviso*, atributos y criterios ergonómicos

12.3.3 Atributos de Structured Text Entry

De este mecanismo de usabilidad se han utilizado en el experimento sus tres formas de uso: *Especificar el tipo de visualización del campo de entrada*; *Definición de máscaras*; *Valores por defecto*.

La primera de las formas de uso, tiene como objetivo especificar el formato del campo de entrada para facilitar la labor de introducir datos al usuario. El analista debe poder elegir el tipo de campo de entrada que se adapte mejor a cada una de las situaciones. La Tabla 12.7 muestra los atributos y los criterios ergonómicos que tienen el mismo objetivo que esta forma de uso.

Atributo Modelo Usabilidad Abrahamo	Atributo de la ISO/IEC 9126-1	Criterio Ergonómico Bastien y Scapin
Minimización de acciones	Ninguno	Acciones mínimas
Familiaridad de conceptos	Claridad de los elementos de la interfaz	Compatibilidad
Prevención de errores	Validación de elementos de entrada	Protección a errores

Tabla 12.7 Relación entre *Especificar el tipo de visualización del campo de entrada*, atributos y criterios ergonómicos

La forma de uso *Definición de máscaras* tiene como objetivo el evitar que el usuario introduzca datos con un formato no válido. Este objetivo coincide con el objetivo de los atributos y los criterios ergonómicos representados en la Tabla 12.8.

Atributo Modelo Usabilidad Abrahamo	Atributo de la ISO/IEC 9126-1	Criterio Ergonómico Bastien y Scapin
Validación de datos	Validación de elementos de entrada	Protección a errores
Prevención de errores	Claridad de los elementos de la interfaz	Incitación

Calidad de los mensajes	Claridad del mensaje	Calidad de los mensajes de error
-------------------------	----------------------	----------------------------------

Tabla 12.8 Relación entre *Definición de máscaras*, atributos y criterios ergonómicos

Por último, la forma de uso *Valores por defecto* tiene como objetivo guiar al usuario para que sepa cuál es el formato que debe utilizar en la entrada de datos. Este objetivo coincide con los atributos y los criterios ergonómicos de la Tabla 12.9.

Atributo Modelo Usabilidad Abrahamo	Atributo de la ISO/IEC 9126-1	Criterio Ergonómico Bastien y Scapin
Complejidad de valores iniciales	Complejidad de descripción	Incitación
Prevención de errores	Validación de elementos de entrada	Protección a errores

Tabla 12.9 Relación entre *Valores por defecto*, atributos y criterios ergonómicos

12.3.4 Atributos de Multilevel Help

De las dos formas de uso de *Multilevel help*, se ha implementado en el experimento la *Ayuda dinámica*. El objetivo de esta forma de uso es el de mostrar ayuda dependiendo de la acción que realice el usuario. La Tabla 12.10 muestra los atributos y los criterios ergonómicos que tienen los mismos objetivos que la forma de uso *Ayuda dinámica*.

Atributo Modelo Usabilidad Abrahamo	Atributo de la ISO/IEC 9126-1	Criterio Ergonómico Bastien y Scapin
Distinción por capacitación	Adaptabilidad	Experiencia del usuario
Complejidad de documentación	Complejidad de documentación y ayuda	Ninguno

Tabla 12.10 Relación entre *Ayuda dinámica*, atributos y criterios ergonómicos

En base a los atributos relacionados con cada una de las formas de uso, se ha elaborado el cuestionario que los usuarios respondieron tras interactuar con el sistema. A continuación se explica en detalle las tareas y los cuestionarios utilizados en el experimento.

12.4 Instrumentos utilizados en el experimento

Tal y como se ha comentado en la sección del diseño del experimento (sección 12.2) se han utilizado tres instrumentos: un cuestionario demográfico, cuatro tareas y un cuestionario para capturar la usabilidad percibida por el usuario. En el Anexo II se puede ver en detalle la información recogida mediante estos instrumentos. A continuación se detalla el contenido de cada uno de estos instrumentos en una subsección. Los instrumentos tareas y cuestionarios se explican en una misma subsección porque las preguntas del cuestionario están divididas por tareas. Cada vez que el usuario finaliza una tarea, debe contestar las preguntas del cuestionario referentes a esa tarea.

12.4.1 Cuestionario demográfico

Este cuestionario contiene preguntas para extraer información de los usuarios que forman parte de la evaluación. Las preguntas que contiene este cuestionario son:

- Género
- Edad
- Ocupación
- Si ha utilizado previamente una aplicación Web. Esta pregunta se responde con verdadero o falso
- Si ha utilizado previamente una aplicación generada con OlivaNOVA. Esta pregunta se responde con verdadero o falso

12.4.2 Tareas y cuestionario

Hay un total de 4 tareas con las que se garantiza la interacción del usuario con el sistema. Cada una de estas tareas trata diversas formas de uso. En esta sección se listan las tareas y las formas de uso estudiadas en cada una de ellas.

Tras ejecutar cada una de las tareas, el usuario debe rellenar las preguntas del cuestionario referentes a las formas de uso tratadas en la tarea. Cada forma de uso se mide mediante una serie de atributos de usabilidad relacionados con la funcionalidad de la forma de uso. Cada uno de los atributos de usabilidad relacionados con las formas de uso da lugar a una pregunta del cuestionario. Es posible que algunos atributos de usabilidad se repitan en varias formas de uso. Los atributos que se repiten, dan lugar a preguntas distintas dentro del cuestionario, ya que su objetivo es distinto, dependiendo de la forma de uso sobre la que midan la usabilidad.

Por último, es importante remarcar que el cuestionario consiste en una escala Likert de 5 puntos. Cada pregunta del cuestionario se redacta como una sentencia en afirmativo y otra en negativo. El usuario debe marcar en la escala la casilla que está más cerca de su opinión. La Tabla 12.11 muestra un ejemplo de una pregunta del cuestionario dividida en una sentencia en afirmativo y otra en negativo. Las posibilidades de respuesta del usuario son las siguientes:

- Si el usuario marca la primera casilla, es que está totalmente de acuerdo con la sentencia en afirmativo.
- Si el usuario marca la segunda casilla, es que está bastante de acuerdo con la sentencia en afirmativo.
- Si el usuario marca la tercera casilla es que no lo tiene claro o está indeciso, término medio.
- Si el usuario marca la cuarta casilla es que está bastante de acuerdo con la sentencia en negativo.
- Si el usuario marca la quinta casilla es que está totalmente de acuerdo con la sentencia en negativo.

Todas las preguntas del cuestionario se responden de la misma forma.


Sentencia en afirmativo	Respuesta	Sentencia en negativo
Los campos que he rellenado validan los datos antes de ejecutar la acción de crear una nueva cuenta bancaria		La acción de crear una nueva cuenta bancaria se ejecuta sin validar los campos que he rellenado

Tabla 12.11 Ejemplo de pregunta del cuestionario.

A continuación se presentan las tareas y las preguntas del cuestionario que se rellenan tras cada tarea.

12.4.2.1 Tarea 1

La tarea 1 es: “Crea un coche nuevo en la oficina ‘AVIS VALENCIA’ (oficina ya existente) con características similares al coche que tengas (si no tienes coche inventa los datos de uno). En la casilla tarifa pon la cantidad que consideres.”

Esta tarea trata las siguientes formas de uso: *Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)*; *Especificar el tipo de visualización del campo de entrada (WU_STE1)*; *Valores por defecto (WU_STE3)*. Cada una de estas formas de uso lleva asociada un conjunto de preguntas del cuestionario, una pregunta por cada atributo de usabilidad relacionado con la forma de uso.

La Tabla 12.12 presenta la relación que hay entre las preguntas del cuestionario, cada uno de los atributos de usabilidad y las formas de uso. Para cada pregunta, se muestra la sentencia en afirmativo y en negativo. La sentencia en afirmativo se ha identificado mediante el acrónimo A y la sentencia en negativo mediante el acrónimo N.

Forma de uso	Atributo de Usabilidad	Pregunta del cuestionario (dividida en dos sentencias)
Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)	Realimentación informativa	(P1A) Estoy seguro de si la acción de crear coche se ha ejecutado o no correctamente
		(P1N) Dudo si la acción de crear coche se ha ejecutado o no correctamente
	Calidad de los mensajes	(P2A) Entiendo de manera clara y rápida si las acciones se han ejecutado o no correctamente y porqué
		(P2N) Me cuesta entender de manera clara y rápida si la acción de crear coche se ha ejecutado o no correctamente y porqué
Especificar el tipo de visualización del campo de entrada (WU_STE1)	Minimización de acciones	(P3A) Creo que el tipo de los campos a rellenar es el óptimo para ahorrarme trabajo
		(P3N) Creo que el tipo de los campos a rellenar se podría mejorar para ahorrarme trabajo
	Familiaridad de conceptos	(P4A) El tipo de los campos a rellenar me es familiar
		(P4N) El tipo de los campos a rellenar me es desconocido
	Prevención de errores	(P5A) El tipo de los campos a rellenar me ayuda a evitar errores

Forma de uso	Atributo de Usabilidad	Pregunta del cuestionario (dividida en dos sentencias)
		(P5N) El tipo de los campos a rellenar no me ayuda a evitar errores
Valores por defecto (WU_STE3)	Complejidad de valores iniciales	(P6A) El sistema rellena los campos automáticamente siempre que sea posible
		(P6N) El sistema no rellena los campos automáticamente aun cuando podría ser posible
	Prevención de errores	(P7A) Los campos rellenados automáticamente facilitan mi trabajo
		(P7N) Los campos rellenados automáticamente no facilitan mi trabajo

Tabla 12.12 Preguntas del cuestionario para la tarea 1

En esta tarea, el atributo de usabilidad *Prevención de errores* está repetido en la forma de uso WU_STE1 y WU_STE3. A pesar de la repetición, las preguntas del cuestionario son distintas porque el objetivo de cada forma de uso es distinto. Repeticiones como ésta se dan entre los atributos de otras tareas.

12.4.2.2 Tarea 2

La tarea 2 es: “Crea una nueva cuenta bancaria para el cliente Manuel García Ruíz (cliente ya existente). En el campo saldo debes poner el valor 12000,999. Este valor es un error, ya que la cantidad de euros sólo puede tener dos decimales como máximo. Si no te acepta este valor, corrígelo por un valor correcto. Rellena el resto de campos con los datos que consideres adecuados”.

Esta tarea trata las siguientes formas de uso: *Definición de máscaras (WU_STE2)*; *Ayuda dinámica (WU_MH1)*. La Tabla 12.13 presenta la relación que hay entre las preguntas del cuestionario, cada uno de los atributos de usabilidad y las formas de uso. Para cada pregunta se muestra la sentencia en afirmativo y en negativo. La sentencia en afirmativo se ha identificado mediante el acrónimo A y la sentencia en negativo mediante el acrónimo N.

Forma de uso	Atributo de Usabilidad	Pregunta del cuestionario (dividida en dos sentencias)
Definición de máscaras (WU_STE2)	Validación de datos	(P8A) Los campos que he rellenado validan los datos antes de ejecutar la acción de crear una nueva cuenta bancaria
		(P8N) La acción de crear una nueva cuenta bancaria se ejecuta sin validar los campos que he rellenado
	Prevención de errores	(P9A) Los campos que he rellenado me previenen de errores antes de ejecutar las acciones correspondientes
		(P9N) Los campos que he rellenado para crear una nueva cuenta bancaria no me previenen de errores antes de ejecutar la acción
	Calidad de los mensajes	(P10A) Los mensajes de error en la validación de los campos que he rellenado me explican de manera sencilla cómo corregir el error

Forma de uso	Atributo de Usabilidad	Pregunta del cuestionario (dividida en dos sentencias)
		(P10N) Los mensajes de error en la validación de los campos que he rellenado no explican de manera sencilla cómo corregir el error
Ayuda dinámica (WU_MH1)	Distinción por capacitación	(P11A) La ayuda del sistema se adapta a mi nivel de conocimiento
		(P11N) La ayuda del sistema no se puede adaptar a mi nivel de conocimiento
	Compleitud de la documentación	(P12A) La ayuda del sistema contiene la información que necesito
		(P12N) La ayuda del sistema contiene información irrelevante

Tabla 12.13 Preguntas del cuestionario para la tarea 2

12.4.2.3 Tarea 3

La tarea 3 es: “Los alquileres por más de un mes son raros, de ahí que si un cliente hace una reserva para más de un mes, es muy probable que se deba a un error. Reserva un alquiler para el cliente Manuel García Ruiz con el coche que has creado en la tarea 1 (el coche se creó para la oficina AVIS VALENCIA). La reserva será desde el día 1 de septiembre al 15 de octubre”.

Esta tarea sólo trata una forma de uso: *Mensaje de aviso (WU_W1)*. La Tabla 12.14 presenta la relación que hay entre las preguntas del cuestionario, cada uno de los atributos de usabilidad y la forma de uso. Para cada pregunta, se muestra la sentencia en afirmativo y en negativo. La sentencia en afirmativo se ha identificado mediante el acrónimo A y la sentencia en negativo mediante el acrónimo N.

Forma de uso	Atributo de Usabilidad	Pregunta del cuestionario (dividida en dos sentencias)
Mensaje de aviso (WU_W1)	Realimentación informativa	(P13A) El sistema me previene de posibles errores antes de ejecutar la acción
		(P13N) El sistema no me alerta de posibles errores antes de ejecutar la acción
	Calidad de los mensajes	(P14A) Los mensajes que me previenen de acciones que pueden ser erróneas son claros
		(P14N) Los mensajes que me previenen de acciones que pueden ser erróneas son difíciles de entender
	Prevención de errores	(P15A) Los mensajes que avisan de acciones que pueden ser erróneas son útiles para prevenir errores
		(P15N) Los mensajes que avisan de acciones que pueden ser erróneas son innecesarios

Tabla 12.14 Preguntas del cuestionario para la tarea 3

12.4.2.4 Tarea 4

La tarea 4 es: “Intenta poner en venta el coche con matrícula 8862CPF con el precio que consideres más justo a fecha de hoy ¿Qué ocurre?” Al intentar poner en venta el coche, el usuario se da cuenta de que el coche tiene alquileres pendientes para el futuro y por tanto, no se puede poner a la venta.

Esta tarea sólo trata una forma de uso: *Mostrar el estado de las acciones (WU_SSF3)*. La Tabla 12.15 presenta la relación que hay entre las preguntas del cuestionario, cada uno de los atributos de usabilidad y la forma de uso. Para cada pregunta, se muestra la sentencia en afirmativo y en negativo. La sentencia en afirmativo se ha identificado mediante el acrónimo A y la sentencia en negativo mediante el acrónimo N.

Forma de uso	Atributo de Usabilidad	Pregunta del cuestionario (dividida en dos sentencias)
Mostrar el estado de las acciones (WU_SSF3)	Realimentación informativa	(P16A) El sistema me informa de las acciones que no puedo lanzar
		(P16N) El sistema no me informa de las acciones que no puedo lanzar
	Calidad de los mensajes	(P17A) Los mensajes me informan de manera clara porqué no puedo lanzar una acción
		(P17N) Los mensajes que me informan de porqué no puedo lanzar la acción no son claros
	Prevención de errores	(P18A) El sistema me previene de aquellas acciones que no puedo ejecutar antes de que cometa un error
		(P18N) El sistema no me previene de aquellas acciones que no puedo ejecutar y acabo cometiendo un error

Tabla 12.15 Preguntas del cuestionario para la tarea 4

12.4.2.5 Cuestionario final

Una vez el usuario ha realizado las 4 tareas y ha rellenado las preguntas del cuestionario relacionadas con cada una de las tareas, debe rellenar un

último cuestionario. Este cuestionario final trata de capturar la opinión del usuario sobre la usabilidad de la aplicación Web en general. Por tanto, las preguntas no están relacionadas con ninguna forma de uso ni con ningún atributo de usabilidad. Las preguntas son las mostradas en la Tabla 12.16. Para cada pregunta, se muestra la sentencia en afirmativo y en negativo. La sentencia en afirmativo se ha identificado mediante el acrónimo A y la sentencia en negativo mediante el acrónimo N.

Pregunta del cuestionario (dividida en dos setencias)
(P19A) La aplicación es fácil de utilizar
(P19N) La aplicación es difícil de utilizar
(P20A) Recomendaría esta aplicación a otras personas que trabajen en el alquiler de coches
(P20N) Nunca recomendaría esta aplicación a otras personas que trabajen en el alquiler de coches
(P21A) De manera general, estoy satisfecho con la aplicación
(P21N) De manera general, no estoy nada satisfecho con la aplicación

Tabla 12.16 Preguntas generales del cuestionario

A continuación se presentan las dos aplicaciones Web con las que interactúan los sujetos del experimento. Independientemente del tipo de aplicación con la que interactúen, las tareas y las preguntas del cuestionario eran las mismas.

12.5 Aplicaciones Web utilizadas en el experimento

Esta sección describe las dos aplicaciones Web utilizadas en el experimento. Ambas aplicaciones implementan el mismo sistema de alquiler

de coches, pero una aplicación no utiliza formas de uso y la otra sí. La comparativa entre ambas aplicaciones se va a centrar en las diferencias de interacción que existen a la hora de ejecutar las tareas que forman parte del experimento. A continuación se describen las diferencias que existen entre ambas aplicaciones según las tareas del experimento.

12.5.1 Tarea 1: Crear coche

La tarea 1 del experimento consiste en crear un coche con los datos del coche del usuario en una oficina de alquiler ya existente. La Figura 12.2 muestra la ventana para crear un coche sin utilizar formas de uso. Al finalizar la ejecución del servicio crear coche, el sistema sólo informa al usuario si el coche no se ha creado, pero no le confirma que se haya creado correctamente.

The screenshot shows a web application window titled "AlquilerCoches". At the top, there is a navigation menu with the following items: Alquiler, Factura, Cliente, Coche, and Gestión oficinas. Below the menu, the page title is "Crear coche". The form contains the following fields:

- Marca: Text input field.
- Matrícula: Text input field.
- Combustible: Text input field.
- Oficina: Text input field with a magnifying glass icon.
- Caballos: Text input field.
- Aire acondicionado: Dropdown menu.
- Modelo: Text input field.
- Número de puertas: Text input field.
- Color: Text input field.
- Radio CD: Dropdown menu.
- Tarifa por día: Text input field.

At the bottom right of the form, there are two buttons: "Aceptar" and "Cancelar".

Figura 12.2 Ventana para crear coche sin utilizar formas de uso

Por otro lado, se puede apreciar cómo los campos para la entrada de datos no están adaptados para facilitar la labor del usuario. El campo *Combustible* sólo acepta como posibles valores *Diésel* y *Gasolina*, por lo que un *RadioButton* o una lista desplegable (*ListBox*) sería el tipo de campo más idóneo. Además, el campo *Oficina* podría ser una lista desplegable. El campo tipo lupa abre una nueva ventana con un conjunto de instancias de oficina en la que el usuario debe seleccionar la que desee. En este caso, la lista de oficinas no es lo suficientemente grande como para necesitar una

nueva ventana y se podría listar en un ListBox, facilitando la labor del usuario.

Por último, otra de las desventajas de esta ventana es que no presenta valores por defecto, a pesar de que la mayoría de coches nuevos tienen Radio CD, aire acondicionado y son de motor diésel. Con todo esto, los usuarios que interactúen con esta aplicación deben valorar bastante negativamente las preguntas de la Tabla 12.12.

Los usuarios que interactúan con la aplicación que utilizaba las formas de uso emplean una interfaz como la mostrada en la Figura 12.3. Esta aplicación informa con un mensaje en la cabecera de la página Web si el sistema se ha ejecutado correctamente (Figura 12.4). Esta funcionalidad se extrae de la forma de uso *Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)*.

Además, los tipos de los campos de entrada de datos están optimizados en esta ventana: hay un ListBox para campos que tienen entre 3 y 15 posibles valores; hay RadioButtons para campos que tienen sólo 2 posibles valores. Esta optimización se deriva de la forma de uso *Especificar el tipo de visualización del campo de entrada (WU_STE1)*.

The screenshot shows a web application window titled "AlquilerCoches" with a "Ver Ayuda" link. The breadcrumb navigation is "Alquiler > Factura > Cliente > Coche > Gestión oficinas". The main heading is "Crear coche (Crea un coche nuevo)". The form contains the following fields:

- Marca: Text input field.
- Matrícula: Text input field.
- Combustible: Radio buttons for "Diésel" (selected) and "Gasolina".
- Radio CD: Radio buttons for "Tiene" (selected) and "No Tiene".
- Tarifa por día: Text input field.
- Modelo: Text input field.
- Número de puertas: Text input field.
- Color: Text input field.
- Oficina: Dropdown menu showing "Avis Valencia".
- Caballos: Text input field.
- Aire acondicionado: Radio buttons for "Tiene" (selected) and "No Tiene".

At the bottom right, there are "Aceptar" and "Cancelar" buttons.

Figura 12.3 Ventana para crear coche utilizando formas de uso

Por último, presenta como valores por defecto aquellos que son más comunes entre los coches nuevos, tal como propone la forma de uso *Valores por defecto (WU_STE3)*. Con todo esto, los usuarios que interactúan con esta aplicación deben valorar bastante positivamente las preguntas de la Tabla 12.12.

El informar si el servicio se ha ejecutado o no correctamente, la optimización del tipo de campos de entrada y la existencia de valores por defecto, se ha aplicado a todas las tareas del experimento.

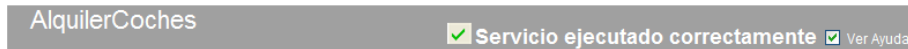


Figura 12.4 Mensaje para confirmar que el servicio se ha ejecutado correctamente

12.5.2 Tarea 2: Crear cuenta bancaria

La tarea 2 del experimento consiste en crear una nueva cuenta bancaria para un cliente ya existente. La Figura 12.5 representa la ventana desde la que se crea una cuenta bancaria en la aplicación que no utiliza las formas de uso. Ninguno de los campos de entrada de datos tiene una máscara, lo cual hace que no se garantice la introducción de datos en un formato correcto. Por ejemplo, el número de cuenta debe ser cuatro dígitos numéricos seguidos por un guión y diez dígitos numéricos más (Ejemplo, 1111-2222222222). A la hora de realizar la tarea, se les indica a los usuarios que introduzcan mal el campo *saldo disponible*, poniendo el valor 12000,999. Al no haber una máscara que prohíba la existencia de más de dos decimales, la aplicación no muestra ningún tipo de error al introducir ese saldo. De esta manera el usuario puede comprender que el sistema no está validando correctamente sus datos de entrada.

Además, esta ventana no tiene ningún tipo de ayuda dinámica. Por ejemplo, podría explicar cómo manejar el botón calendario. Con todo esto, los usuarios que interactúen con esta aplicación deben valorar bastante negativamente las preguntas de la Tabla 12.13.

Figura 12.5 Ventana para crear cuenta bancaria sin utilizar formas de uso

Por otro lado, la Figura 12.6 muestra una captura de ventana de la aplicación que utiliza formas de uso. En este caso, aplicando la forma de uso *Definición de máscaras (WU_STE2)*, el sistema avisa en el caso de que el usuario introduzca mal el formato de los campos *número de cuenta* y *saldo disponible*. Así el usuario puede corregir los valores conforme a lo explicado en el mensaje de error.

Además, el usuario puede utilizar la ayuda dinámica. Esta ayuda se activa y desactiva mediante el checkbox que hay en la cabecera de la página Web, llamado *Ver ayuda*. La ayuda está disponible en todas las tareas. Esta funcionalidad se deriva de la forma de uso *Ayuda dinámica (WU_MH1)*.

Figura 12.6 Ventana para crear cuenta bancaria utilizando formas de uso

Con todo esto, los usuarios que interactúen con esta aplicación deben valorar bastante positivamente las preguntas de la Tabla 12.13.

12.5.3 Tarea 3: Hacer una reserva de alquiler

La tarea 3 consiste en hacer una reserva de un alquiler durante más de un mes (del 1 de septiembre al 15 de octubre). La reserva para tanto tiempo se da en pocas ocasiones y, por tanto, es muy probable que se deba a un error en la introducción de las fechas. La Figura 12.7 muestra una captura de ventana de la interfaz para hacer una reserva de alquiler en la aplicación que no utiliza las formas de uso. Esta aplicación ejecuta el servicio *Reservar alquiler de coche* sin dar ningún aviso al usuario sobre un posible error en caso de que el periodo de reserva sea superior a un mes. Los usuarios que interactúen con esta aplicación deben valorar bastante negativamente las preguntas de la Tabla 12.14

Figura 12.7 Ventana para hacer una reserva de alquiler sin utilizar formas de uso

Por otro lado, la ventana la Figura 12.8 representa la aplicación que incorpora la forma de uso *Mensaje de aviso (WU_W1)*. En esta aplicación, cuando el usuario solicita una reserva de un alquiler por más de un mes se le muestra un mensaje de aviso para informar de un posible error (Figura 12.9). El usuario al leer este mensaje puede aceptar o rechazar la ejecución de ese servicio. Los usuarios que interactúen con esta aplicación deben valorar bastante positivamente las preguntas de la Tabla 12.14.

AlquilerCoches Ver Ayuda Administrar Cambio de contraseña

Alquiler ▶ Factura ▶ Cliente ▶ Coche ▶ Gestión oficinas ▶

Alquilar vehículo (Hace una reserva de un coche para alquilarlo)

Oficina:

Cliente:

Coche:

Fecha prevista de recogida: Pulse sobre el botón calendario. Use las flechas para moverte entre años. Para buscar años lejanos, pulse 1 vez sobre el año que muestra el calendario (búsqueda del año 1 a 1) o pulse 2 veces (búsqueda por decenas).

Fecha prevista de devolución: Pulse sobre el botón calendario. Use las flechas para moverte entre años. Para buscar años lejanos, pulse 1 vez sobre el año que muestra el calendario (búsqueda del año 1 a 1) o pulse 2 veces (búsqueda por decenas).

Figura 12.8 Ventana para hacer una reserva de alquiler utilizando formas de uso

AlquilerCoches Ver Ayuda

Alquiler ▶ Factura ▶ Cliente ▶ Coche ▶ Gestión oficinas ▶

Aviso

Las fechas seleccionadas para la reserva son por un periodo superior a 30 días. ¿Está seguro de la elección?
Pulse Aceptar para confirmar la reserva del alquiler o Cancelar para rechazar la acción

Figura 12.9 Mensaje de aviso para reservas de alquiler con más de un mes de duración

12.5.4 Tarea 4: Poner coche en venta

La tarea 4 consiste en poner el coche con matrícula 8862 CPF en venta y ver qué ocurre. El resultado tras la ejecución de esta tarea es que el usuario debe entender que el coche no se puede poner en venta porque aún tiene alquileres pendientes para el futuro. La Figura 12.10 muestra la ventana para poner un coche en venta en la aplicación que no utiliza formas de uso. En esta aplicación, como el coche indicado para poner en venta tiene alquileres pendientes, se muestra un mensaje de error como el de la Figura 12.11 cuando el usuario pulsa el botón *Aceptar* (lanza el servicio). Los

usuarios que interactúen con esta aplicación deben valorar bastante negativamente las preguntas de la Tabla 12.15.

The screenshot shows the 'AlquilerCoches' application window. At the top, there is a navigation bar with tabs for 'Alquiler', 'Factura', 'Cliente', 'Coche', and 'Gestión oficinas'. Below this, the title 'Poner en venta' is displayed. The main area contains a form with the following elements:

- A label 'Coche' followed by a yellow search icon and a magnifying glass icon.
- A label 'Precio' followed by an empty text input field.
- A label 'Fecha de puesta en venta' followed by a date picker showing '28/07/2009' and a calendar icon.
- At the bottom right, there are two buttons: 'Aceptar' and 'Cancelar'.

Figura 12.10 Ventana para poner un coche en venta sin utilizar formas de USO

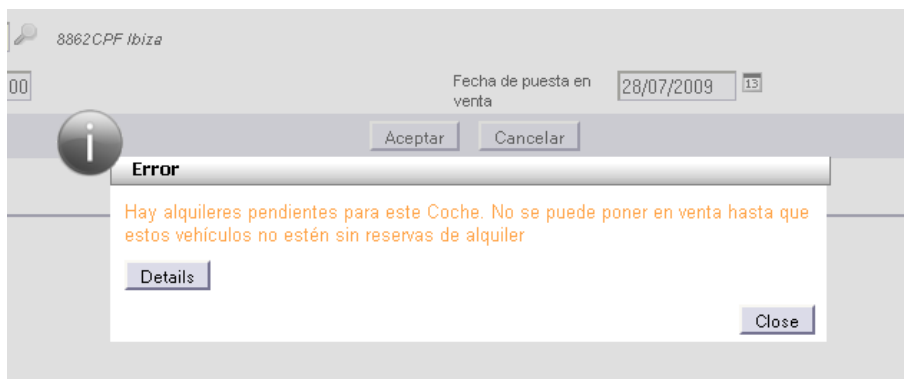


Figura 12.11 Mensaje de error cuando el usuario solicita poner un coche en venta habiendo alquileres pendientes

Por otro lado, la Figura 12.12 muestra la ventana para poner un coche en venta en la aplicación que utiliza la forma de uso *Mostrar el estado de las acciones (WU_SSF3)*. En este caso, cuando el usuario selecciona un coche para poner en venta teniendo alquileres pendientes, se inhabilita el botón *Aceptar* y además se muestra un mensaje de aviso, *Hay alquileres pendientes para este coche*. De esta manera se evita que el usuario cometa un error. Los usuarios que interactúen con esta aplicación deben valorar bastante positivamente las preguntas de la Tabla 12.15.

The screenshot shows a web application interface for 'AlquilerCoches'. At the top, there is a navigation bar with 'Alquiler', 'Factura', 'Cliente', 'Coche', and 'Gestión oficinas'. On the right, there are links for 'Administrado', 'Ver Ayuda', and 'Cambio de contraseña'. Below the navigation bar, the main heading is 'Poner en venta (Cambia el estado de un coche de alquilable a estar en venta)'. The form contains the following fields: 'Coche' with a dropdown menu showing '2' and a speech bubble icon next to '8862CPF Ibiza'; 'Precio' with a text input field containing '1.000,00'; and 'Fecha de puesta en venta' with a date picker showing '28/07/2009'. To the right of the date picker is a tooltip with instructions: 'Pulse sobre el botón calendario Use las flechas para moverte entre años Para buscar años lejanos, pulse 1 vez sobre el año que muestra el calendario (búsqueda del año 1 a 1) o pulse 2 veces (búsqueda por decenas)'. At the bottom of the form, there is a status bar that says 'Hay alquileres pendientes para este coche' and two buttons: 'Aceptar' and 'Cancelar'.

Figura 12.12 Ventana para poner un coche en venta utilizando formas de uso

A continuación se va a realizar un análisis detallado de los datos obtenidos con ambas aplicaciones Web.

12.6 Análisis de datos

Esta sección analiza los datos obtenidos en el experimento. El primer paso en el análisis de datos es el de eliminar de los datos capturados en el experimento aquellos que son anómalos. El total de participantes del experimento son 66 y se han detectado 2 datos anómalos, por lo que la muestra que se ha tomado para analizar son 64 usuarios. Estos 2 usuarios se han tomado como anómalos porque en ambos casos se han rellenado todas las casillas con el mismo valor y además han terminado la prueba en un tiempo inferior al del resto de usuarios, a pesar de ser en ambos casos usuarios novatos. Esto hace pensar que rellenaron la encuesta sin completar las tareas que se les pedían, como un mero trámite.

En base a estos 64 usuarios se estudia si las hipótesis nulas definidas en el diseño experimental son o no ciertas. En caso de que no sean ciertas, se comprobará si las hipótesis alternativas se cumplen. El total de hipótesis incluidas en el diseño experimental son dos. Una hipótesis afirma que la satisfacción de los usuarios que usan aplicaciones Web con formas de uso es igual a la satisfacción de los usuarios que usan aplicaciones Web sin formas de uso (H_{10}). La otra hipótesis afirma que la eficiencia de los usuarios que usan aplicaciones Web con formas de uso es igual a la

eficiencia de los usuarios que usan aplicaciones Web sin formas de uso (H_{20}). A continuación, se estudia cada una de estas hipótesis en una subsección distinta.

12.6.1 Hipótesis sobre la satisfacción del usuario (H_{10}):

La satisfacción del usuario se estudia en base al cuestionario que cada usuario rellena tras finalizar cada tarea. Los resultados se analizan en base a tres conceptos estadísticos: la comparación entre las medias, ANOVA o análisis de varianza y gráficos box and whiskers. La comparación de las medias proporciona una primera visión de los resultados obtenidos, el ANOVA se utiliza para demostrar la significancia de los dos variables independientes (experiencia con el uso de aplicaciones desarrolladas con OlivaNOVA y uso de las formas de uso) con respecto a la satisfacción. Por último, los gráficos box and whiskers se han utilizado para ver gráficamente la mediana y los cuartiles para los dos factores. En base a estos tres análisis estadísticos se determinará si la hipótesis H_{10} es o no cierta.

12.6.1.1 Estudio de las medias

Las medias proporcionan un resultado preliminar antes de entrar en estudios más complejos como el ANOVA. La Figura 12.13 muestra una comparativa de las medias aritméticas de las respuestas a las preguntas del cuestionario entre los cuatro tipos de usuarios estudiados. Cada uno de los tipos de usuarios del experimento se corresponde con una serie del gráfico. Los acrónimos de las series del gráfico son los siguientes:

- **RespuestaECF:** Son las respuestas de los expertos en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que incorporaba las formas de uso.
- **RespuestaNCF:** Son las respuestas de los novatos (no expertos) en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que incorporaba formas de uso.

- **RespuestaESF:** Son las respuestas de los expertos en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que no incorporaba formas de uso.
- **RespuestaNSF:** Son las respuestas de los novatos en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que no incorporaba formas de uso.

El eje Y de la Figura 12.13 representa la media aritmética del valor de la satisfacción de los usuarios. Los números de este eje representan lo siguiente:

- El valor 1 representa que el usuario está totalmente de acuerdo con la sentencia en afirmativo de la pregunta.
- El valor 2 representa que el usuario está bastante de acuerdo con la sentencia en afirmativo de la pregunta.
- El valor 3 representa que el usuario no lo tiene claro o está indeciso, término medio.
- El valor 4 representa que el usuario está bastante de acuerdo con la sentencia en negativo de la pregunta.
- El valor 5 representa que el usuario está totalmente de acuerdo con la sentencia en negativo de la pregunta.

Todas las preguntas del cuestionario contienen en su sentencia en afirmativo características buenas de la usabilidad, por lo tanto, cuanto más bajos sean los valores en las respuestas de los usuarios, mejor será su valoración con respecto a la usabilidad de la aplicación Web.

Por otro lado, el eje X representa las 21 preguntas que componen el cuestionario. Estas preguntas son las explicadas en la Tabla 12.12, la Tabla 12.13, la Tabla 12.14, la Tabla 12.15, la Tabla 12.16.

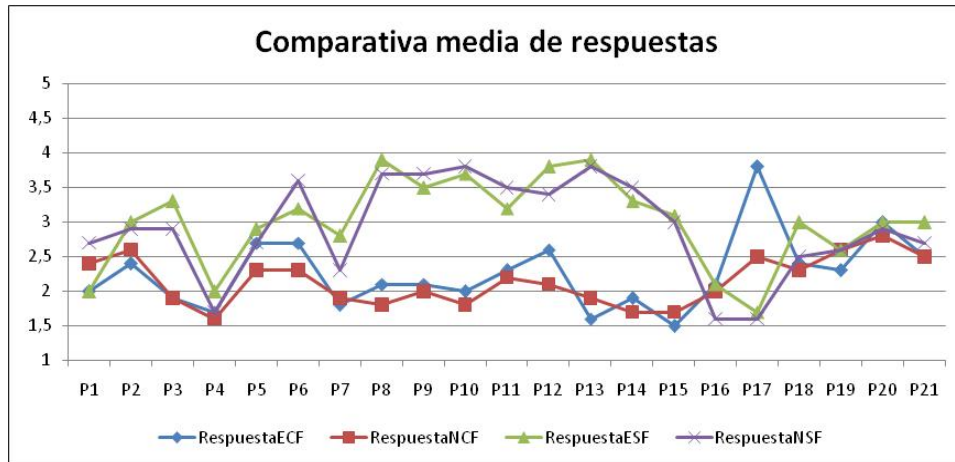


Figura 12.13 Comparativa de las medias aritméticas de las respuestas entre los cuatro tipos de usuarios

A continuación se analizan detalladamente los datos para cada una de las preguntas:

- P1 (Tabla 12.12):** No existe mucha diferencia entre los cuatro tipos de interacción estudiados. Las valoraciones de los expertos son mejores que las valoraciones de los novatos porque saben que las aplicaciones OlivaNOVA nunca avisan si la ejecución ha finalizado con éxito, es decir, sólo avisan si se produce algún error. De manera general, todos los usuarios han valorado positivamente que la aplicación informa sobre el éxito o fracaso de la ejecución de servicios (independientemente de la facilidad con la que obtienen esta información).
- P2 (Tabla 12.12):** En este caso tampoco existe mucha diferencia entre los cuatro tipos de interacción estudiados. Los grupos de usuarios que interactúan con la aplicación con formas de uso han valorado esta pregunta más positivamente que los que interactúan con la aplicación sin formas de uso. Este hecho se puede deber a que en la aplicación sin formas de uso se debe consultar el listado de coches para estar seguro que el coche se ha creado satisfactoriamente, mientras que en la aplicación con formas de uso,

esta búsqueda no es necesaria gracias a un mensaje de aviso (Figura 12.4).

- **P3 (Tabla 12.12):** En la evaluación de esta pregunta hay una diferencia importante entre aquellos usuarios que han interactuado con la aplicación que incorpora formas de uso con respecto a la que no las incorpora. Los que han interactuado con la aplicación que incorpora las formas de uso han valorado esta pregunta más positivamente. Esto se puede deber a que la aplicación adapta los tipos de campo de entrada de datos a los más idóneos para cada dato. Tal y como se puede apreciar en la Figura 12.3, se han utilizado RadioButtons y ListBoxes que no se han utilizado en la aplicación sin formas de uso (Figura 12.2).
- **P4 (Tabla 12.12):** No existe diferencia entre los cuatro grupos de usuarios estudiados. Los tipos de campos de entrada utilizados en ambas aplicaciones son conocidos por todos los usuarios, tanto expertos como novatos.
- **P5 (Tabla 12.12):** Existe una mínima diferencia entre el grupo de los novatos que interactúan con la aplicación que incorpora formas de uso con respecto a los otros tres grupos. Este grupo ha valorado positivamente el hecho de que el tipo de campo de entrada de datos le ayude a evitar errores. Esto se debe a que los RadioButton y ListBoxes utilizados en la aplicación con formas de uso, sólo permiten marcar datos válidos (Figura 12.3). Por el contrario, el tipo de campo de entrada de la aplicación sin formas de uso (Figura 12.2) permite introducir valores no válidos que producen errores en el sistema.
- **P6 (Tabla 12.12):** Existe una pequeña diferencia entre los resultados de los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la que no las incorpora. La aplicación sin formas de uso no rellena automáticamente ningún campo, es decir, no utiliza valores por defecto (Figura 12.2). En cambio, la aplicación que incorpora formas de uso rellena automáticamente varios de los campos de entrada (Figura 12.3). Esto explica que los usuarios de la aplicación con

formas de uso hayan valorado esta pregunta más positivamente que los usuarios de la aplicación sin formas de uso.

- **P7 (Tabla 12.12):** Existe una pequeña diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la que no incorpora. Los usuarios que trabajan con la aplicación con valores por defecto valoran más positivamente la pregunta sobre si los valores por defecto facilitan la labor del usuario. De los usuarios que interactúan con la aplicación sin valores por defecto, echan más en falta los valores por defecto los expertos en OlivaNOVA que los novatos.
- **P8 (Tabla 12.13):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la aplicación que no las incorpora. Los usuarios que interactúan con la aplicación con formas de uso valoran esta pregunta más positivamente porque les informa del error cometido al introducir un saldo erróneo (Figura 12.6). Es decir, esta aplicación valida los datos antes de ejecutar el servicio de crear una cuenta bancaria. En cambio, la aplicación sin formas de uso no realiza esta validación de los datos y ejecuta el servicio sin informar al usuario sobre el error.
- **P9 (Tabla 12.13):** Los usuarios que interactúan con la aplicación que incorpora formas de uso valoran mucho más positivamente esta pregunta que los que interactúan con la aplicación que no incorpora formas de uso. La aplicación con formas de uso previene del error antes de que se lance la ejecución del servicio *crear cuenta bancaria* (Figura 12.6). En cambio, la aplicación sin formas de uso (Figura 12.5) no previene del error. Este hecho explica la gran diferencia de valoración entre los grupos de usuarios que interactúan con formas de uso y los que interactúan sin ellas.
- **P10 (Tabla 12.13):** Los usuarios que interactúan con la aplicación que incorpora formas de uso evalúan mucho más positivamente esta pregunta que los que interactúan con la aplicación que no incorpora formas de uso. La aplicación con formas de uso muestra un mensaje explicativo que alerta del error a la hora de introducir los

datos y cuál debe ser el formato correcto de los datos a introducir. Este mensaje se muestra en los campos *saldo disponible* y *número de cuenta* (Figura 12.6). Todos los usuarios visualizan el mensaje del campo *saldo disponible*, ya que la tarea 2 consiste en introducir el saldo 12000,999, que es erróneo. Por otro lado, el campo *número de cuenta* lo rellena libremente cada usuario. En caso de que el formato del número de cuenta sea el no requerido, se le muestra el mensaje con información sobre cuál debe ser el formato correcto a utilizar. El uso de mensajes de aviso en los dos campos explican que los usuarios con formas de uso evalúen esta pregunta más favorablemente.

- **P11 (Tabla 12.13):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la aplicación que no las incorpora. Según la evaluación de los usuarios, la ayuda mostrada de manera dinámica en la aplicación con formas de uso (Figura 12.6) se adapta muy bien al nivel de conocimiento del usuario.
- **P12 (Tabla 12.13):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la que no las incorpora. Los usuarios que interactúan con la aplicación con formas de uso valoran más positivamente que la aplicación incluya mensajes de ayuda que son útiles para el usuario. La ayuda dinámica está mejor valorada en los usuarios novatos que en los expertos, ya que los novatos están menos familiarizados con los entornos OlivaNOVA y, por tanto, requieren de ayuda a la hora de interactuar con la aplicación.
- **P13 (Tabla 12.14):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la que no las incorpora. La aplicación con formas de uso muestra un mensaje de aviso antes de hacer la reserva del alquiler por más de un mes (Figura 12.9). De ahí, que los usuarios que han interactuado con la aplicación con formas de uso hayan valorado esta pregunta más positivamente que los otros.

- **P14 (Tabla 12.14):** Los usuarios que interactúan con la aplicación que incorpora formas de uso valoraron mucho más positivamente esta pregunta que los que interactúan con la aplicación que no incorpora formas de uso. Esto se debe a que la aplicación sin formas de uso no tiene ningún mensaje explicativo sobre el posible error que se comete al hacer la reserva por más de un mes. En cambio, la aplicación con formas de uso tiene un mensaje de advertencia claro (Figura 12.9) para todo tipo de usuarios (novatos y expertos).
- **P15 (Tabla 12.14):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la que no las incorpora. Los usuarios que interactúan con la aplicación con formas de uso valoran esta pregunta más positivamente que los que interactúan con la aplicación sin formas de uso. Esta diferencia se debe a que la aplicación sin formas de uso no muestra ningún mensaje de aviso, y por tanto, no se previene al usuario de un posible error.
- **P16 (Tabla 12.15):** En esta pregunta no existe prácticamente ninguna diferencia entre los cuatro grupos de usuarios estudiados. Todos los usuarios la valoran muy positivamente. Esto se debe a que tanto la aplicación que incorpora formas de uso (Figura 12.12) como la que no las incorpora (Figura 12.11) muestran un mensaje informando de que el servicio *poner en venta* no se puede ejecutar.
- **P17 (Tabla 12.15):** Existe una pequeña diferencia entre los usuarios que interactúan con formas de uso y los que interactúan sin formas de uso. En este caso, los que interactúan con formas de uso evalúan más negativamente esta pregunta que los que interactúan con la aplicación sin formas de uso. Esta diferencia se hace más evidente en el grupo de expertos que interactúan con formas de uso, que han evaluado esta pregunta mucho más negativamente que el resto de grupos de usuarios. Este resultado se puede deber a que el mensaje de advertencia es más claro en la aplicación sin formas de uso (Figura 12.11) que en la aplicación con formas de uso (Figura 12.12). Posiblemente, la implementación de esta forma de uso no ha sido la óptima y el mensaje de texto que se muestra

en la aplicación con formas de uso podría ser más explicativo. Un mensaje como el de la Figura 12.11 hubiera sido más adecuado.

- **P18 (Tabla 12.15):** No existe mucha diferencia entre los cuatro grupos de usuarios. El único grupo que se destaca un poco del resto es el de usuarios expertos que interactúan con la aplicación sin formas de uso. Los sujetos de este grupo valoran la pregunta más negativamente que el resto de usuarios. Esto se debe a que los usuarios expertos son los únicos que se dan cuenta de que el mensaje de aviso no es un mensaje de error. En cambio, los usuarios novatos toman el mensaje de aviso como un mensaje de error, de ahí que no haya diferencia entre novatos que interactúan con formas de uso y novatos que interactúan sin formas de uso.
- **P19 (Tabla 12.16):** Existe una diferencia muy pequeña entre los usuarios expertos que interactúan con la aplicación que incorpora formas de uso con respecto al resto de grupos de usuarios. Los usuarios expertos con formas de uso valoran esta pregunta un poco más positivamente con respecto al resto de usuarios. Esto se debe a que los usuarios expertos conocen las aplicaciones OlivaNOVA y por tanto son conscientes de las mejoras que aportan las formas de uso con respecto a lo ya existente en este tipo de aplicaciones. En cambio, los usuarios novatos, al no conocer otras aplicaciones OlivaNOVA no saben si las formas de uso facilitan o no la labor del usuario con respecto a lo que ya existe hoy en día en las aplicaciones OlivaNOVA.
- **P20 (Tabla 12.16):** No existe ninguna diferencia entre los cuatro grupos de usuarios estudiados. Todos los grupos han valorado esta pregunta con un término medio. Por lo tanto, las formas de uso estudiadas no han favorecido para que los usuarios recomienden esta aplicación a otras personas.
- **P21 (Tabla 12.16):** Existe una diferencia muy pequeña entre los usuarios expertos que interactúan sin formas de uso con respecto al resto de usuarios. Los usuarios expertos que interactúan sin formas de uso valoran esta pregunta más negativamente que el resto. Esto se debe a que los usuarios expertos son conscientes de las mejoras

que se han incorporado en la aplicación generada, mientras que los usuarios novatos, al no conocer las aplicaciones OlivaNOVA, no pueden comparar las mejoras incorporadas con respecto a lo que ya existe hoy en día en este tipo de aplicaciones.

A partir del análisis por preguntas mediante medias, se puede concluir que la P20 (la cuestión que obtiene la opinión del usuario sobre si recomendaría la aplicación a otros usuarios) es totalmente irrelevante para nuestro experimento, ya que los cuatro tipos de sujetos estudiados obtienen el mismo valor. Por lo tanto, esta pregunta se elimina en futuros análisis.

Una vez finalizado un análisis detallado para cada una de las preguntas del cuestionario, se hace un análisis detallado por formas de uso. Para ello, se han agrupado las preguntas del cuestionario que pertenecen a una misma forma de uso. La agrupación ha consistido en calcular la media aritmética entre las preguntas que componen cada una de las formas de uso estudiadas. Las preguntas que componen cada una de las formas de uso se pueden consultar en la Tabla 12.12, la Tabla 12.13, la Tabla 12.14, la Tabla 12.15, la Tabla 12.16.

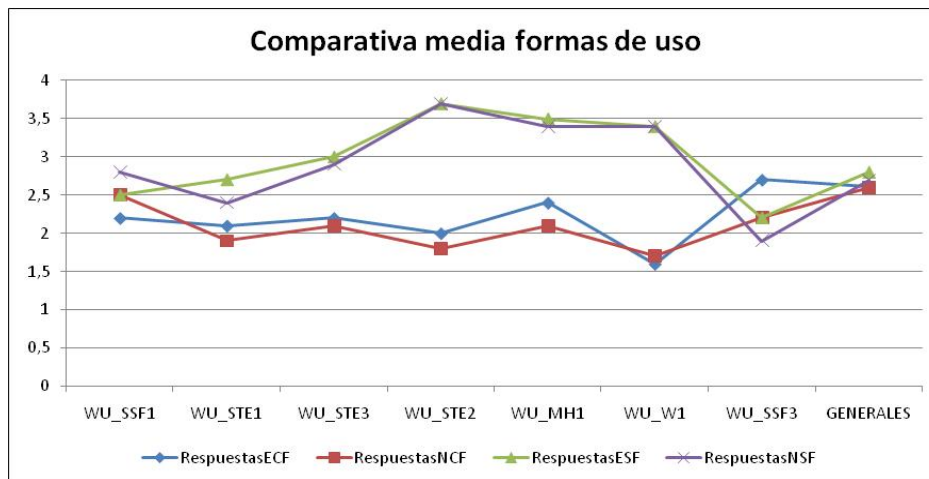


Figura 12.14 Comparativa de las medias aritméticas de las formas de uso entre los cuatro tipos de usuarios

La Figura 12.14 muestra el gráfico con las medias aritméticas calculadas a partir de las respuestas de los usuarios. El eje Y representa la media

aritmética de la forma de uso calculada a partir de las respuestas a las preguntas del cuestionario. El eje X representa las formas de uso tratadas en el experimento. La interpretación de los valores del eje Y es la misma que la utilizada en la Figura 12.13.

A continuación se comentan los resultados obtenidos para cada una de las formas de uso estudiadas:

- **Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1):** Existe una diferencia apreciable entre el grupo de usuarios expertos que interactúa con formas de uso y el grupo de usuarios novatos que interactúa sin formas de uso. Los usuarios expertos que interactúan con formas de uso han valorado WU_SSF1 más positivamente que los novatos que interactúan con formas de uso. En cambio, los usuarios expertos que interactúan sin formas de uso y los novatos con formas de uso han valorado WU_SSF1 con el mismo valor. Por lo tanto, esta forma de uso es valorada más positivamente por los expertos que por los novatos.
- **Especificar el tipo de visualización del campo de entrada (WU_STE1):** Esta forma de uso es valorada positivamente en los grupos de usuarios que interactúan con formas de uso, aunque su diferencia con los usuarios que interactúan sin formas de uso no es muy grande. WU_STE1 está mejor valorada entre los usuarios novatos que entre los expertos.
- **Valores por defecto (WU_STE3):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso con respecto a la aplicación que no las incorpora. Los usuarios que utilizan la aplicación con WU_STE3 valoran esta forma de uso muy positivamente, independientemente de si son usuarios novatos o expertos. Por lo tanto, esta forma de uso es muy valorada entre los usuarios, independientemente de su experiencia.
- **Definición de máscaras (WU_STE2):** De entre todas las formas de uso estudiadas, WU_STE2 es la que obtiene mayor diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la que no las incorpora. Los

usuarios que interactúan con WU_STE2 evalúan esta forma de uso muy positivamente, independientemente de si se trata de usuarios novatos o expertos. Por lo tanto, esta forma de uso es muy valorada entre los usuarios, independientemente de su experiencia.

- **Ayuda dinámica (WU_MH1):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la aplicación que no las incorpora. Los usuarios que interactúan con WU_MH1 valoran esta forma de uso muy positivamente. Además, es importante resaltar que los usuarios novatos que interactúan con WU_MH1 valoran esta forma de uso más positivamente que los usuarios expertos que también interactúan con ella. Por lo tanto, esta forma de uso es muy apreciada por los usuarios, especialmente por los novatos.
- **Mensaje de aviso (WU_W1):** Existe una gran diferencia entre los usuarios que interactúan con la aplicación que incorpora formas de uso y los que interactúan con la aplicación que no las incorpora. Los usuarios que interactúan con la aplicación que incorpora WU_W1 valoran esta forma de uso muy positivamente, independientemente de si el usuario es experto o novato. Por lo tanto, esta forma de uso es muy apreciada tanto en los usuarios novatos como en los expertos.
- **Mostrar el estado de las acciones (WU_SSF3):** Existe una diferencia entre los usuarios expertos que interactúan con formas de uso y los usuarios novatos que no interactúan con formas de uso. Los expertos evalúan WU_SSF3 de manera más negativa que los usuarios novatos que no interactúan con esta forma de uso. Los otros dos grupos de usuarios, novatos con formas de uso y expertos sin formas de uso, evalúan WU_SSF3 con el mismo valor. Por lo tanto, WU_SSF3 no ha sido valorada positivamente, especialmente por los usuarios expertos.
- **Cuestiones de usabilidad generales:** No existe una diferencia apreciable entre los cuatro grupos de usuarios estudiados. Sólo hay una mínima diferencia entre los usuarios que interactúan con formas de uso y los que interactúan sin ellas. Por lo tanto, la percepción

general de usabilidad del sistema es prácticamente la misma para los cuatro grupos de usuarios.

La Tabla 12.17 muestra un resumen de los tipos de usuario que valoran más cada una de las formas de uso.

Mecanismo	Forma de uso	Tipo de usuario
System Status Feedback	WU_SSF1	Expertos con formas de uso
	WU_SSF3	Novatos sin formas de uso
Structured Text Entry	WU_STE1	Novatos con formas de uso
	WU_STE2	Novatos con formas de uso
	WU_STE3	Expertos y novatos con formas de uso
Multilevel Help	WU_MH1	Novatos con formas de uso
	WU_W1	Expertos y novatos con formas de uso
Usabilidad en general		No hay diferencia

Tabla 12.17 Tipos de usuario que mejor valoran las formas de uso

12.6.1.2 Estudio del ANOVA

El análisis de la varianza (ANOVA) es un método estadístico que permite interpretar objetivamente los efectos de uno o varios factores en una variable. Se trata de un procedimiento de verificación de la igualdad de las medias poblacionales de los grupos que pueden formarse al clasificar la población objeto de estudio, con arreglo a las categorías del factor o factores considerados. En este caso, la variable independiente es la satisfacción.

La satisfacción se ha capturado con una escala Likert de 5 puntos, por lo que es una variable cualitativa. ANOVA sólo trabaja con variables cuantitativas, por lo tanto, es necesario convertir la satisfacción a cuantitativa. Para hacer esta conversión es necesario agrupar las preguntas que se han hecho al usuario. La agrupación ha consistido en sumar el valor obtenido en cada una de las preguntas. Por ejemplo, si queremos agrupar tres preguntas y el usuario ha marcado el valor 2 para la pregunta 1, el valor 5 para la pregunta 2 y el valor 3 para la pregunta 3, la agrupación tendría el valor de 10. Es importante aclarar que el número de elementos que forman cada agrupación no tiene porqué ser el mismo para todos los elementos estudiados en el ANOVA. Es decir, puede haber grupos de 2, 3 y 4 elementos formando parte del mismo estudio. Se han hecho 2 agrupaciones distintas:

1. Se han agrupado las preguntas por la forma de uso que miden. Cada una de las preguntas del cuestionario está relacionada con una única forma de uso.
2. Se han agrupado las preguntas dependiendo de si el atributo de usabilidad que se mide con esa pregunta es subjetivo u objetivo. Es decir, se diferencia entre atributos cuyo valor depende de la manera de pensar del usuario que ha rellenado la encuesta y atributos cuyo valor es independiente de la manera de pensar del usuario. Las formas de uso deben afectar directamente a los atributos objetivos e indirectamente a los subjetivos, según las guías de la IPO. El grupo de atributos objetivos está compuesto por: realimentación informativa; completitud de valores iniciales; prevención de errores; validación de datos; prevención de errores; distinción por capacitación. Dentro del grupo de atributos subjetivos están: calidad de los mensajes; minimización de acciones; familiaridad de conceptos; prevención de errores; completitud de la documentación.

A continuación se muestran los resultados del ANOVA para los dos tipos de agrupación.

1. Agrupación por formas de uso:

Se va a trabajar con ocho variables dependientes (todas ellas representan la satisfacción del usuario), es decir, las siete formas de uso estudiadas más las preguntas generales sobre la usabilidad del sistema. Estas variables se han llamado así:

- SUMRespuestaWU_SSF1: Suma de las respuestas a las cuestiones que miden la forma de uso *Informar del éxito o fracaso en la ejecución del servicio*.
- SUMRespuestaWU_STE1: Suma de las respuestas a las cuestiones que miden la forma de uso *Especificar el tipo de visualización del campo de entrada*.
- SUMRespuestaWU_STE3: Suma de las respuestas a las cuestiones que miden la forma de uso *Valores por defecto*.
- SUMRespuestaWU_STE2: Suma de las respuestas a las cuestiones que miden la forma de uso *Definición de máscaras*.
- SUMRespuestaWU_MH1: Suma de las respuestas a las cuestiones que miden la forma de uso *Ayuda dinámica*.
- SUMRespuestaWU_W1: Suma de las respuestas a las cuestiones que miden la forma de uso *Mensaje de aviso*.
- SUMRespuestaWU_SSF3: Suma de las respuestas a las cuestiones que miden la forma de uso *Mostrar el estado de las acciones*.
- SUMRespuestaGeneral: Suma de las respuestas a las cuestiones sobre la usabilidad en general. No se ha tenido en cuenta las respuestas a la cuestión 20 de la Tabla 12.16 porque se ha demostrado anteriormente que no aporta ningún valor al estudio.

Para poder calcular el ANOVA es necesario asumir los siguientes supuestos [Visauta, 1997]:

- **Normalidad:** Los valores de la variable dependiente (satisfacción) se distribuyen normalmente en cada una de las k poblaciones.
- **Independencia:** Las muestras son independientes y extraídas de forma aleatoria de poblaciones determinadas.
- **Igualdad de varianzas:** Las poblaciones de procedencia de las k muestras extraídas tienen todas la misma varianza.

Prueba de Kolmogorov-Smirnov para una muestra

		SUMRespues taWU_SSF1	SUMRespues taWU_STE1	SUMRespues taWU_STE3	SUMRespues taWU_STE2
N		66	66	66	66
Parámetros normales ^a	Media	5,17	6,91	5,20	8,64
	Desviación típica	2,704	2,854	2,047	3,995
Diferencias más extremas	Absoluta	,167	,142	,183	,124
	Positiva	,167	,142	,115	,124
	Negativa	-,156	-,091	-,183	-,117
Z de Kolmogorov-Smirnov		1,356	1,155	1,486	1,009
Sig. asintót. (bilateral)		,050	,139	,024	,261

a. La distribución de contraste es la Normal.

Figura 12.15 Prueba K-S para WU_SSF1, WU_STE1, WU_STE3, WU_STE2

Para demostrar la normalidad de la agrupación de las preguntas por formas de uso, se ha utilizado la prueba de normalidad de Kolmogorov-Smirnov (K-S). El resultado tras aplicar este test en la herramienta SPSS es el mostrado en la Figura 12.15 y Figura 12.16.

Prueba de Kolmogorov-Smirnov para una muestra

		SUMRespues taWU_MH1	SUMRespues taWU_W1	SUMRespues taWU_SSF3	SUMRespues taGeneral
N		66	66	66	66
Parámetros normales ^a	Media	5,80	7,85	6,82	5,33
	Desviación típica	2,355	4,073	3,521	2,342
Diferencias más extremas	Absoluta	,148	,161	,152	,185
	Positiva	,148	,161	,152	,185
	Negativa	-,109	-,117	-,139	-,100
Z de Kolmogorov-Smirnov		1,206	1,308	1,234	1,504
Sig. asintót. (bilateral)		,109	,065	,095	,022

a. La distribución de contraste es la Normal.

Figura 12.16 Prueba K-S para WU_MH1, WU_W1, WU_SSF3, Generales

El resultado de la prueba K-S se mide con el valor obtenido en el apartado significancia asintótica. Dependiendo del valor de este campo, se acepta o

se rechaza la hipótesis de la prueba K-S. La hipótesis nula de esta prueba afirma que la variable dependiente sigue una distribución normal. La Tabla 12.18 muestra cómo se debe interpretar el valor obtenido en el campo significancia. En base a dicha tabla, se puede afirmar que se cumple de manera general la hipótesis nula, y que por tanto **la satisfacción sigue una distribución normal**.

Significancia>0.1	Se acepta completamente la hipótesis nula
0,05<=Significancia<0.1	Se acepta parcialmente la hipótesis nula
0,01<=Significancia<0,05	Se acepta en término medio la hipótesis nula
0,001<=Significancia<0,01	Se rechaza parcialmente la hipótesis nula
Significancia<0,001	Se rechaza completamente la hipótesis nula

Tabla 12.18 Interpretación de la significancia

La segunda de las condiciones para poder aplicar el ANOVA es la independencia de las muestras. Para ello se puede utilizar la prueba de los residuos de Durbin-Watson. Se suele aceptar que las muestras son independientes si el estadístico da un valor entre 1,5 y 2,5. Los resultados para cada una de las variables dependientes al aplicar el estadístico Durbin-Watson se puede ver en la Tabla 12.19. En vista a estos resultados se puede afirmar que **las muestras son independientes**.

Variable dependiente	Estadístico Durbin-Watson
SUMRespuestaWU_SSF1	2,48
SUMRespuestaWU_STE1	2,08
SUMRespuestaWU_STE3	2,19
SUMRespuestaWU_STE2	1,89

SUMRespuestaWU_MH1	1,85
SUMRespuestaWU_W1	1,76
SUMRespuestaWU_SSF3	2,12
SUMRespuestaGeneral	2,2

Tabla 12.19 Prueba de independencia de Durbin-Watson para la satisfacción

La tercera de las condiciones para poder aplicar ANOVA es la igualdad de varianzas. Para ello se va a aplicar la prueba de Levene sobre homogeneidad de varianzas. El resultado de esta prueba se mide con el valor obtenido en el apartado significancia. Dependiendo del valor de este campo, se acepta o se rechaza la hipótesis de la prueba de Levene. La hipótesis nula de esta prueba afirma que existe homogeneidad de varianzas. El resultado de la prueba de Levene se muestra en la Figura 12.17. La interpretación de la significancia es la mostrada en la Tabla 12.18. Por tanto, se concluye que se acepta la hipótesis nula que afirma que **hay homogeneidad de varianzas**.

Prueba de homogeneidad de varianzas

	Estadístico de Levene	gl1	gl2	Sig.
SUMRespuestaWU_SSF1	,914	1	64	,343
SUMRespuestaWU_STE1	,434	1	64	,512
SUMRespuestaWU_STE3	,795	1	64	,376
SUMRespuestaWU_STE2	,082	1	64	,775
SUMRespuestaWU_MH1	,077	1	64	,783
SUMRespuestaWU_W1	,151	1	64	,698
SUMRespuestaWU_SFE3	,169	1	64	,682
SUMRespuestaGeneral	3,198	1	64	,078

Figura 12.17 Prueba de Levene para la satisfacción

Una vez demostrado que se cumplen las tres condiciones, se va a aplicar el ANOVA a las ocho variables dependientes que representan la satisfacción

del usuario. El análisis de la varianza permite contrastar la hipótesis nula de que las medias de K poblaciones ($K > 2$) son iguales, frente a la hipótesis alternativa de que por lo menos una de las poblaciones difiere de las demás en cuanto a su valor esperado [Visauta, 1997]. Hay una hipótesis nula por cada una de las formas de uso estudiadas, y otra para las preguntas generales de usabilidad (ocho hipótesis en total).

La Figura 12.18 muestra el resultado del ANOVA tras aplicar el factor experiencia en el uso de aplicaciones OlivaNOVA a las ocho variables dependientes con las que se estudia la satisfacción. A partir del valor de la columna significancia y en base a la Tabla 12.18, se puede afirmar que la hipótesis nula de que las medias de las poblaciones estudiadas son iguales se puede aceptar. Es decir, **no existe una relación entre la satisfacción del usuario (formas de uso y preguntas generales) y la experiencia previa en el uso de aplicaciones OlivaNOVA.**

ANOVA						
		Suma de cuadrados	gl	Media cuadrática	F	Sig.
SUMRespuestaWU_SSF1	Inter-grupos	6,371	1	6,371	,870	,355
	Intra-grupos	468,795	64	7,325		
	Total	475,167	65			
SUMRespuestaWU_STE1	Inter-grupos	5,523	1	5,523	,675	,414
	Intra-grupos	523,932	64	8,186		
	Total	529,455	65			
SUMRespuestaWU_STE3	Inter-grupos	,485	1	,485	,114	,737
	Intra-grupos	271,955	64	4,249		
	Total	272,439	65			
SUMRespuestaWU_STE2	Inter-grupos	1,091	1	1,091	,067	,796
	Intra-grupos	1036,182	64	16,190		
	Total	1037,273	65			
SUMRespuestaWU_MH1	Inter-grupos	1,939	1	1,939	,346	,558
	Intra-grupos	358,500	64	5,602		
	Total	360,439	65			
SUMRespuestaWU_W1	Inter-grupos	,189	1	,189	,011	,916
	Intra-grupos	1078,295	64	16,848		
	Total	1078,485	65			
SUMRespuestaWU_SSFE3	Inter-grupos	27,273	1	27,273	2,242	,139
	Intra-grupos	778,545	64	12,165		
	Total	805,818	65			
SUMRespuestaGeneral	Inter-grupos	,008	1	,008	,001	,971
	Intra-grupos	356,659	64	5,573		
	Total	356,667	65			

Figura 12.18 ANOVA de la satisfacción con el factor experiencia

La Figura 12.19 muestra el resultado del ANOVA tras aplicar el factor formas de uso a las ocho variables dependientes. A partir del valor de la columna significancia, y en base a la Tabla 12.18, se puede concluir que la hipótesis nula de que las medias de las poblaciones estudiadas son iguales se puede aceptar sólo para algunas de las formas de uso (WU_SSF1, WU_SSF3, Respuestas Generales). Para el resto de formas de uso, existe una relación entre la satisfacción del usuario y las formas de uso utilizadas, es decir, **se cumple la hipótesis alternativa que afirma que existe relación entre la forma de uso estudiada y la satisfacción del usuario**. Concretamente, existe relación entre satisfacción y las siguientes formas de uso: WU_STE1, WU_STE3, WU_STE2, WU_MH1, WU_W1

ANOVA						
		Suma de cuadrados	gl	Media cuadrática	F	Sig.
SUMRespuestaWU_SSF1	Inter-grupos	6,682	1	6,682	,913	,343
	Intra-grupos	468,485	64	7,320		
	Total	475,167	65			
SUMRespuestaWU_STE1	Inter-grupos	44,182	1	44,182	5,827	,019
	Intra-grupos	485,273	64	7,582		
	Total	529,455	65			
SUMRespuestaWU_STE3	Inter-grupos	45,833	1	45,833	12,945	,001
	Intra-grupos	226,606	64	3,541		
	Total	272,439	65			
SUMRespuestaWU_STE2	Inter-grupos	458,727	1	458,727	50,745	,000
	Intra-grupos	578,545	64	9,040		
	Total	1037,273	65			
SUMRespuestaWU_MH1	Inter-grupos	94,561	1	94,561	22,762	,000
	Intra-grupos	265,879	64	4,154		
	Total	360,439	65			
SUMRespuestaWU_W1	Inter-grupos	427,636	1	427,636	42,051	,000
	Intra-grupos	650,848	64	10,170		
	Total	1078,485	65			
SUMRespuestaWU_SSF3	Inter-grupos	21,879	1	21,879	1,786	,186
	Intra-grupos	783,939	64	12,249		
	Total	805,818	65			
SUMRespuestaGeneral	Inter-grupos	2,970	1	2,970	,537	,466
	Intra-grupos	353,697	64	5,527		
	Total	356,667	65			

Figura 12.19 ANOVA de la satisfacción con el factor formas de uso

Fuente	Variable dependiente	Suma de cuadrados tipo III	gl	Media cuadrática	F	Significación
Exp * Usab	SUMRespuestaWU_SSF1	,068	1	,068	,009	,924
	SUMRespuestaWU_STE1	,614	1	,614	,079	,779
	SUMRespuestaWU_STE3	,121	1	,121	,033	,856
	SUMRespuestaWU_STE2	2,455	1	2,455	,265	,609
	SUMRespuestaWU_MH1	,758	1	,758	,178	,674
	SUMRespuestaWU_W1	,614	1	,614	,059	,810
	SUMRespuestaWU_SSF3	,758	1	,758	,062	,804
	SUMRespuestaGeneral	1,280	1	1,280	,225	,637

Figura 12.20 ANOVA de la satisfacción estudiando dos factores: experiencia y usabilidad

Por último, se hace un estudio estadístico ANOVA de la satisfacción con dos factores (experiencia y formas de uso). Esto nos permitirá ver si la interacción de ambos factores afectan o no a la satisfacción del usuario [Visauta, 1998]. La Figura 12.20 muestra el resultado. Al comparar los resultados de la columna significación con la Tabla 12.18 se puede concluir que se acepta la hipótesis nula, es decir, que **no hay relación entre la interacción de los factores experiencia (Exp) y formas de uso (Usab) con la satisfacción del usuario.**

2. Agrupación por atributos subjetivos y objetivos:

Se va a trabajar con 3 variables dependientes que representan la satisfacción del usuario. Estas variables se han llamado del siguiente modo:

- **Objetivas:** Es la suma de las respuestas a las cuestiones derivadas de atributos objetivos. Las formas de uso afectan directamente a los atributos que componen este conjunto.
- **Subjetivas:** Es la suma de las respuestas a las cuestiones derivadas de atributos subjetivos. Las formas de uso afectan indirectamente a los atributos que componen este conjunto, según las guías de usabilidad desarrolladas por la IPO.
- **Total:** Es la suma de todas las respuestas, independientemente de si se derivan de atributos objetivos o subjetivos (excepto las

respuestas relacionadas con la usabilidad en general de la aplicación P19, P20 y P21).

Previo al análisis de datos, se ha comprobado si se cumplían los supuestos de normalidad, independencia e igualdad de varianzas necesarios para calcular el ANOVA. **Los resultados obtenidos con los tres tests muestran que se satisfacen las tres condiciones de normalidad.**

La Figura 12.21 presenta el resultado del ANOVA tras aplicar el factor experiencia a las tres variables dependientes. En vista del valor de la columna significancia y en base a la Tabla 12.18, se puede afirmar que la hipótesis nula de que las medias de las poblaciones estudiadas son iguales es cierta. No existe ninguna relación entre el factor experiencia y los atributos objetivos, subjetivos, ni la suma de todos ellos.

ANOVA						
		Suma de cuadrados	gl	Media cuadrática	F	Sig.
Objetivas	Inter-grupos	,371	1	,371	,013	,910
	Intra-grupos	1844,250	64	28,816		
	Total	1844,621	65			
Subjetivas	Inter-grupos	69,818	1	69,818	,814	,370
	Intra-grupos	5492,545	64	85,821		
	Total	5562,364	65			
Total	Inter-grupos	60,008	1	60,008	,298	,587
	Intra-grupos	12887,523	64	201,368		
	Total	12947,530	65			

Figura 12.21 ANOVA de los distintos tipos de atributos con el factor experiencia

La Figura 12.22 muestra el resultado del ANOVA tras aplicar el factor formas de uso a las tres variables dependientes. A partir del valor de la columna significancia y con la Tabla 12.18, se puede concluir que la hipótesis nula de que las medias de las poblaciones estudiadas son iguales es falsa y se debe aceptar la hipótesis alternativa. La hipótesis alternativa afirma que **existe una relación entre el factor formas de uso y los atributos objetivos, subjetivos y todos los atributos independientemente de su tipo.**

ANOVA

		Suma de cuadrados	gl	Media cuadrática	F	Sig.
Objetivas	Inter-grupos	612,136	1	612,136	31,787	,000
	Intra-grupos	1232,485	64	19,258		
	Total	1844,621	65			
Subjetivas	Inter-grupos	1474,909	1	1474,909	23,094	,000
	Intra-grupos	4087,455	64	63,866		
	Total	5562,364	65			
Total	Inter-grupos	3987,409	1	3987,409	28,481	,000
	Intra-grupos	8960,121	64	140,002		
	Total	12947,530	65			

Figura 12.22 ANOVA de los distintos tipos de atributos con el factor formas de uso

Por lo tanto, los atributos objetivos y subjetivos están relacionados con la usabilidad que percibe el usuario. Se hace un estudio más profundo para ver qué atributos están más relacionados con la usabilidad. Para ello se utilizan los atributos que aparecen repetidos en más de una cuestión para hacer una agrupación por atributo. Sólo hay tres atributos que se repitan en más de una cuestión, y por tanto, que se puedan agrupar: realimentación informativa (objetivo); calidad de los mensajes (subjetivo); prevención de errores (subjetivo). El resto de atributos, al no aparecer repetidos no se pueden agrupar y por tanto no se pueden estudiar con el ANOVA, ya que son datos cualitativos.

ANOVA

		Suma de cuadrados	gl	Media cuadrática	F	Sig.
Realimentacion_informativa	Inter-grupos	1,485	1	1,485	,201	,656
	Intra-grupos	473,682	64	7,401		
	Total	475,167	65			
Calidad_mensajes	Inter-grupos	7,280	1	7,280	,659	,420
	Intra-grupos	706,841	64	11,044		
	Total	714,121	65			
Prevencion_errores	Inter-grupos	7,758	1	7,758	,382	,539
	Intra-grupos	1299,227	64	20,300		
	Total	1306,985	65			

Figura 12.23 ANOVA de los tres atributos estudiados con el factor experiencia

La Figura 12.23 muestra el resultado del ANOVA tras aplicar el factor experiencia a los tres atributos repetidos en más de una cuestión. En base a

la columna significancia y a la Tabla 12.18, se puede observar que **no existe relación entre la experiencia de los usuarios y estos tres atributos.**

ANOVA						
		Suma de cuadrados	gl	Media cuadrática	F	Sig.
Realimentacion_informativa	Inter-grupos	68,015	1	68,015	10,691	,002
	Intra-grupos	407,152	64	6,362		
	Total	475,167	65			
Calidad_mensajes	Inter-grupos	117,333	1	117,333	12,583	,001
	Intra-grupos	596,788	64	9,325		
	Total	714,121	65			
Prevencion_errores	Inter-grupos	292,742	1	292,742	18,472	,000
	Intra-grupos	1014,242	64	15,848		
	Total	1306,985	65			

Figura 12.24 ANOVA de los tres atributos estudiados con el factor experiencia

La Figura 12.24 muestra el resultado del ANOVA tras aplicar el factor formas de uso a los tres atributos repetidos en más de una cuestión. En base a la columna significancia y a la Tabla 12.18, se puede concluir que existe una relación fuerte entre el factor formas de uso y los tres atributos. Esta relación es muy fuerte para el atributo *prevención de errores*, se decreta ligeramente para *calidad de los mensajes*, y se decreta un poco más para *Realimentación informativa*. Por tanto, no existen diferencias entre los distintos tipos de atributos, sean objetivos o subjetivos.

12.6.1.3 Gráficos box and whiskers

Para terminar con el análisis de la satisfacción, se muestran los gráficos box and whiskers para algunas de las variables dependientes incluidas en el cálculo del ANOVA. En esta sección se van a comentar los gráficos para la variable SUMRespuestaWU_SSF1 que ha salido no significativa para los dos factores estudiados y SUMRespuestaWU_W1, que ha salido significativa sólo para el factor formas de uso. El resto de variables dependientes tienen unos gráficos similares a SUMRespuestaWU_SSF1 o SUMRespuestaWU_W1 y por tanto carecen de interés.

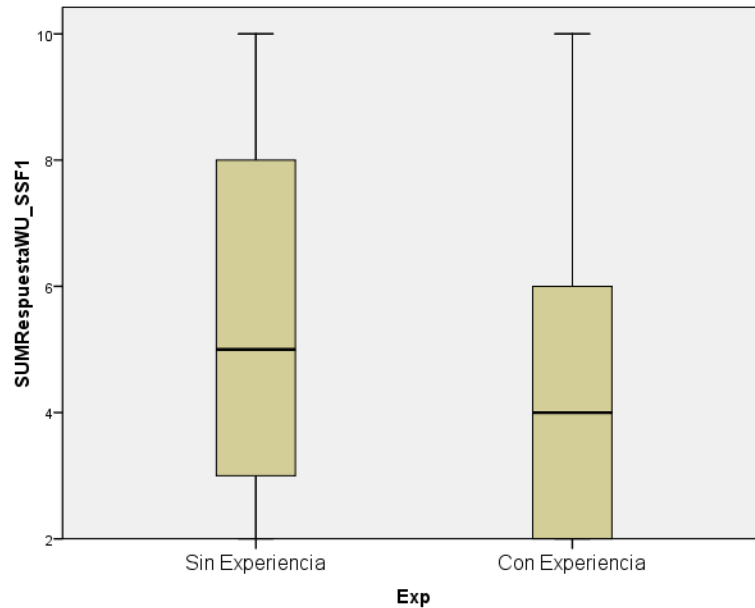


Figura 12.25 Gráfico box and whiskers de SUMRespuestaWU_SSF1 con el factor experiencia

La Figura 12.25 muestra el gráfico del estudio de la variable SUMRespuestaWU_SSF1 con el factor experiencia. Se puede apreciar cómo las medianas (línea dentro de cada caja) son similares tanto para los usuarios con experiencia en OlivaNOVA como para los que no la tienen. Para los usuarios con experiencia, la distribución es simétrica, mientras que para los usuarios sin experiencia es asimétrica positiva. También se observa que la mayoría de los usuarios con experiencia se mueven entre un rango de datos mejor para SUMRespuestaWU_SSF1 (valores pequeños) que la mayoría de los usuarios sin experiencia (valores grandes).

La Figura 12.26 muestra el gráfico del estudio de la variable SUMRespuestaWU_SSF1 con el factor formas de uso. En este caso, las medianas son prácticamente equivalentes tanto para los usuarios que interactuaron con formas de uso como los que interactuaron sin formas de uso. Además, en ambos casos, la distribución es asimétrica positiva. La única diferencia, es que para el caso de los usuarios que interactuaron con

formas de uso, el rango donde se mueve la mayoría de usuarios estudiados es ligeramente mejor (valores pequeños).

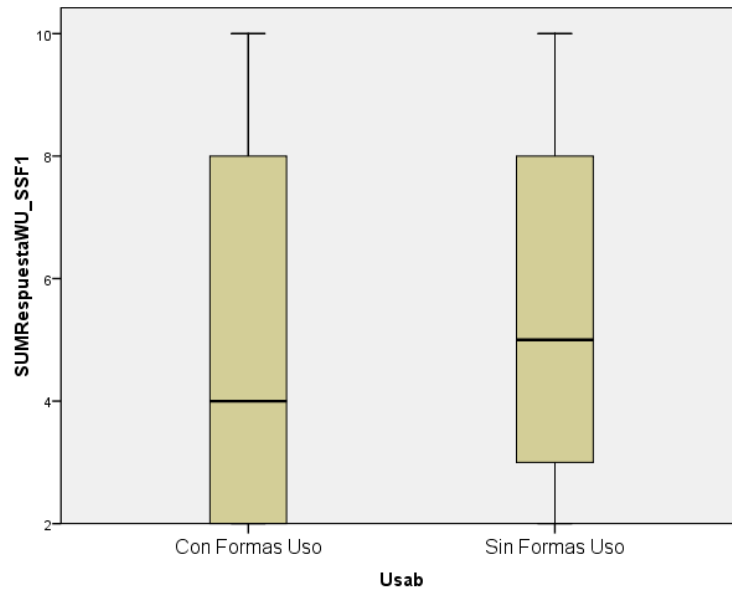


Figura 12.26 Gráfico box and whiskers de SUMRespuestaWU_SSF1 con el factor formas de uso

La Figura 12.27 muestra el gráfico del estudio de la variable SUMRespuestaWU_W1 con el factor experiencia. Las medianas son prácticamente equivalentes tanto para los usuarios con experiencia como sin experiencia. Además, en ambos casos la distribución es asimétrica negativa, más pronunciada en el caso de los usuarios sin experiencia. Como diferencia, cabe destacar que la mayoría de usuarios con experiencia se mueven en un rango ligeramente mejor (valores pequeños) que los usuarios sin experiencia.

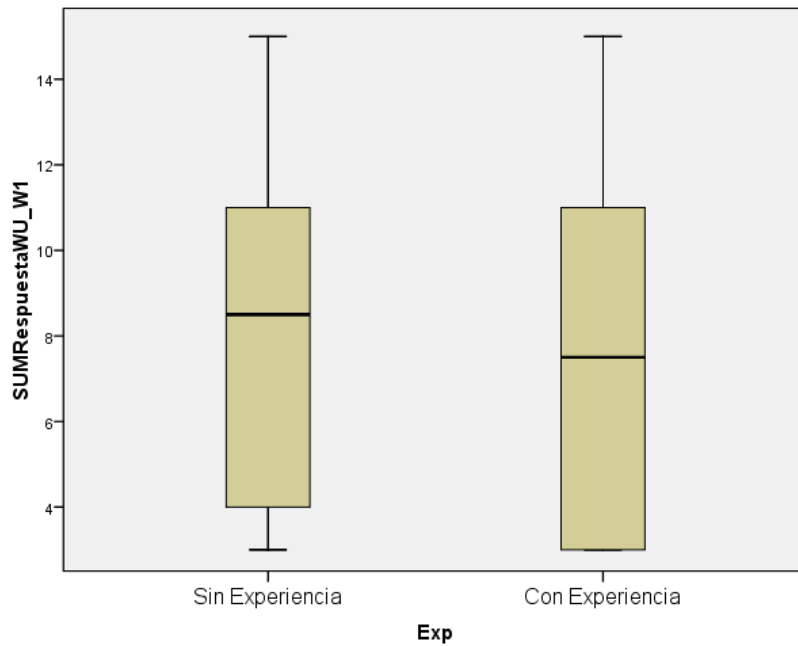


Figura 12.27 Gráfico box and whiskers de SUMRespuestaWU_W1 con el factor experiencia

La Figura 12.28 muestra el gráfico del estudio de la variable SUMRespuestaWU_W1 con el factor formas de uso. Se puede apreciar que existe una diferencia importante entre las medianas de los usuarios que interactúan con formas de uso y los que interactúan sin ellas. La mayoría de valores obtenidos con los usuarios que utilizan la aplicación con formas de uso es mucho mejor que los valores obtenidos con los usuarios que utilizan la aplicación sin las formas de uso. Además, la mediana de los usuarios con formas de uso es asimétrica positiva, mientras que la mediana de los usuarios sin formas de uso es asimétrica negativa. Se han obtenido gráficos similares para todas las formas de uso excepto para WU_SSF1 (como se ha visto anteriormente), WU_SSF3 y las preguntas de usabilidad generales. Para estos tres casos, las medianas no presentan diferencias significativas.

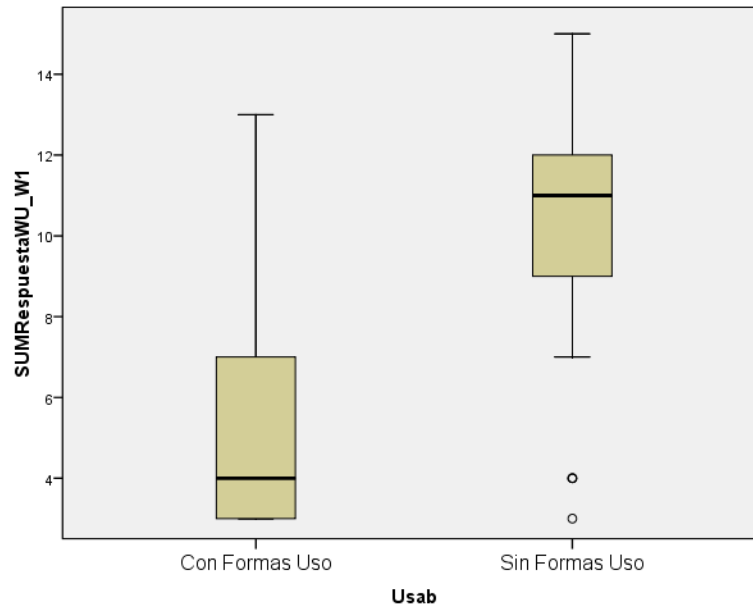


Figura 12.28 Gráfico box and whiskers de SUMRespuestaWU_W1 con el factor formas de uso

12.6.1.4 Conclusiones del análisis de la satisfacción

En base a las distintas agrupaciones que se han realizado para estudiar la satisfacción del usuario con el ANOVA (por formas de uso y por atributos), se puede afirmar que la hipótesis nula H_{1_0} (la satisfacción de los usuarios que usan aplicaciones Web con formas de uso es igual a la satisfacción de los usuarios que usan aplicaciones Web sin formas de uso) se debe rechazar. En base a la comparación de medias, el estudio del ANOVA, y los gráficos box and whiskers, se ha demostrado que la mayoría de las formas de uso (excepto WU_SSF1, WU_SSF3) están relacionadas con la satisfacción del usuario, es decir, que se cumple la hipótesis alternativa H_{1_1} : **La satisfacción de los usuarios que usan aplicaciones Web con formas de uso es mayor que la satisfacción de los usuarios que usan aplicaciones Web sin formas de uso.** Además, se ha demostrado que la satisfacción no se ve afectada por la experiencia previa que tenga el usuario en el uso de aplicaciones OlivaNOVA.

Por otro lado, **no se puede afirmar que exista una relación directa de las formas de uso con la satisfacción general de la aplicación**. Es probable que este hecho se deba a que hay otros muchos atributos de usabilidad distintos a los tratados con las formas de uso que inciden en la usabilidad de la aplicación.

Para finalizar, se ha mostrado que no existe diferencia apreciable en la relación que existe entre la satisfacción del usuario y el tipo de atributo estudiado (objetivo o subjetivo). Los dos tipos de atributos están fuertemente relacionados con la satisfacción del usuario.

A continuación se estudia la hipótesis nula H_2_0 .

12.6.2 Hipótesis sobre la eficiencia del usuario (H_2_0):

La eficiencia del usuario se ha estudiado en base al tiempo empleado para finalizar cada una de las 4 tareas que componen el experimento. Al igual que para el estudio de la satisfacción, los resultados se estudian en base a tres conceptos estadísticos: la comparación entre las medias, ANOVA o análisis de varianza y gráficos box and whiskers.

12.6.2.1 Estudio de las medias

La Figura 12.29 muestra una comparativa de las medias aritméticas del tiempo que cada uno de los cuatro grupos de usuarios tarda en completar cada tarea. Cada uno de los tipos de usuarios del experimento se corresponde con una serie del gráfico. Los acrónimos de las series son los siguientes:

- **MinutosECF:** Son los minutos de los expertos en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que incorpora las formas de uso.
- **MinutosNCF:** Son los minutos de los novatos (no expertos) en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que incorpora formas de uso.

- **MinutosESF:** Son los minutos de los expertos en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que no incorpora formas de uso.
- **MinutosNSF:** Son los minutos de los novatos en aplicaciones OlivaNOVA que han interactuado con la aplicación Web que no incorpora formas de uso.

El eje Y de la Figura 12.29 representa la media aritmética de los minutos que han tardado los cuatro tipos de usuarios en cada una de las cuatro tareas. Por otro lado, el eje X representa las cuatro tareas del experimento.

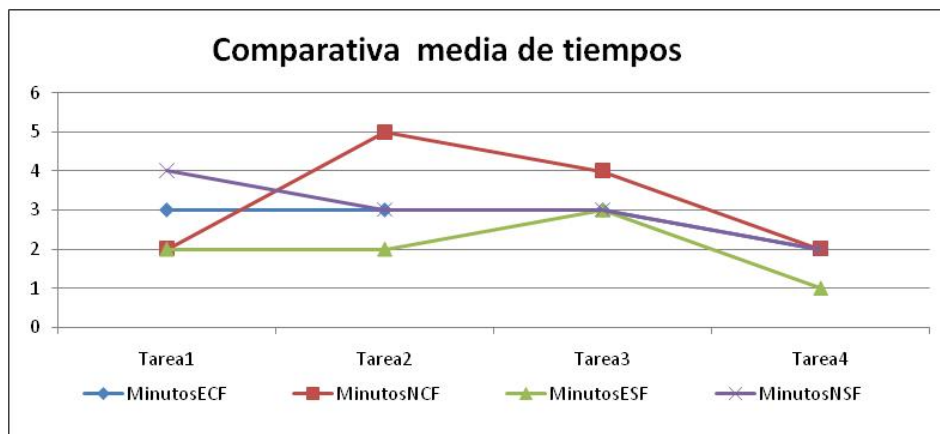


Figura 12.29 Comparativa de las medias aritméticas de los tiempos por tarea entre los cuatro tipos de usuarios

A continuación se hace un análisis de los resultados detallado para cada una de las tareas. Entre paréntesis se indica la tabla donde se explican las formas de uso tratadas en cada una de las tareas:

- **Tarea 1 (Tabla 12.12):** Para esta tarea es importante resaltar que el tiempo que han tardado los usuarios novatos con la aplicación que incorpora las formas de uso es justo la mitad del tiempo que tardan los usuarios novatos con la aplicación que no incorpora las formas de uso. En cuanto a los usuarios expertos, han tardado más tiempo los usuarios con formas de uso que los usuarios sin formas de uso. Esto se puede deber a que los usuarios expertos están habituados

a las aplicaciones OlivaNOVA y por tanto, las variaciones introducidas por las formas de uso (nuevos tipos de campos de entrada) pueden disminuir su eficiencia ya que no les son familiares.

- **Tarea 2 (Tabla 12.13):** Los usuarios expertos con formas de uso tardan un minuto más que los usuarios expertos sin formas de uso. En el caso de los usuarios novatos, los que interactúan en la aplicación con formas de uso tardan dos minutos más que los que interactúan sin formas de uso. Esto se puede deber a que las aplicaciones con formas de uso avisan al usuario del error que ha cometido al introducir los datos (Figura 12.6), y éste tiene que corregirlo. Además, la aplicación con formas de uso tiene ayuda dinámica que el usuario puede leer y por tanto, debe invertir tiempo en dicha lectura. En cambio, la aplicación sin formas de uso no avisa de los errores en la introducción (Figura 12.5) y el usuario no los corrige, por lo que el tiempo a dedicar es inferior en este caso. Además, la aplicación sin formas de uso tampoco dispone de texto de ayuda y no se invierte tiempo en su lectura. Por lo tanto, para esta tarea, la eficiencia es mejor para los usuarios que interactúan sin formas de uso a pesar de que los datos introducidos no sean correctos y no haya ayuda.
- **Tarea 3 (Tabla 12.14):** Los usuarios novatos que interactúan con la aplicación que incorpora formas de uso tardan más que el resto de grupos de usuarios. Esto se puede deber a que la aplicación con formas de uso alerta de un posible error (Figura 12.9), mientras que la aplicación sin formas de uso no alerta. Por tanto, el tiempo de más invertido en los usuarios novatos que interactúan con formas de uso se puede justificar con el tiempo que dedican a entender el mensaje de aviso. Se puede concluir que la eficiencia de los usuarios novatos con formas de uso es peor que los usuarios novatos sin formas de uso. En el caso de los usuarios expertos, el tiempo invertido es idéntico para los usuarios que interactúan con formas de uso y los que interactúan sin ellas.
- **Tarea 4 (Tabla 12.15):** El tiempo de los usuarios expertos sin formas de uso es un minuto inferior al del resto de usuarios. Esta diferencia se debe a que los expertos están familiarizados con

aplicaciones OlivaNOVA y los cambios provocados por la incorporación de las formas de uso añaden nuevas características a las que los usuarios deben dedicar tiempo para entender. En este caso concreto, el mensaje de aviso (Figura 12.12) puede que sea difícil de comprender y de ahí que los expertos dediquen más tiempo para su comprensión. Por tanto, se puede concluir que la eficiencia de los usuarios expertos con formas de uso es peor que la de los usuarios expertos sin formas de uso. En el caso de los usuarios novatos, el tiempo dedicado es el mismo tanto para los que interactúan con formas de uso como para los que interactúan sin ellas.

Una vez hecho un análisis detallado de los tiempos por tareas, se va a hacer un análisis de los tiempos totales que se invirtieron en completar las cuatro tareas. Para ello se ha calculado la media aritmética de los tiempos totales de los usuarios. Es importante destacar que estos tiempos no incluyen el tiempo que los usuarios dedicaron a rellenar las preguntas del cuestionario, es sólo tiempo dedicado a realizar la tarea.

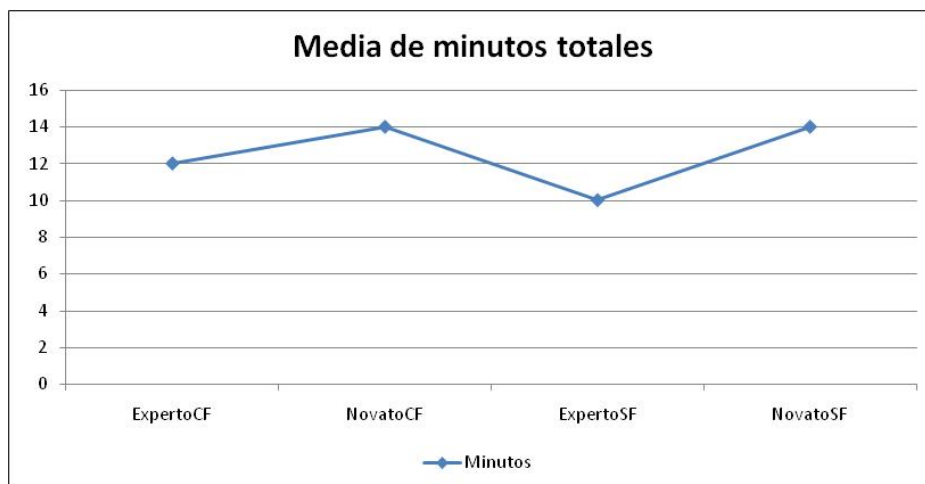


Figura 12.30 Medias aritméticas de los tiempos totales de interacción de los cuatro tipos de usuarios

La Figura 12.30 muestra el gráfico con las medias aritméticas de los tiempos totales. El eje Y representa los minutos invertidos en las cuatro

tareas. El eje X representa los cuatro tipos de usuarios estudiados en el experimento. Son respectivamente: experto con formas de uso; novato con formas de uso; experto sin formas de uso; novato sin formas de uso.

En el gráfico de la Figura 12.30 se puede apreciar que la eficiencia de los novatos es la misma tanto para los que interactúan con formas de uso como para los que interactúan sin ellas. Por lo que respecta a los expertos, son más eficientes los usuarios que interactúan con la aplicación sin formas de uso que los que interactúan con la aplicación con formas de uso. La diferencia de tiempo entre los dos tipos de usuarios expertos es muy pequeña (2 minutos).

12.6.2.2 Estudio del ANOVA

En este caso, la variable independiente es el tiempo invertido para cada una de las cuatro tareas por separado y para su suma. El tiempo ya es una variable cuantitativa, por lo que no es necesario hacer ninguna conversión para aplicar el ANOVA. Se trabaja con cinco variables dependientes que representan el tiempo. Las variables son las siguientes:

- Tiempo_T1: Segundos invertidos para realizar la tarea 1.
- Tiempo_T2: Segundos invertidos para realizar la tarea 2.
- Tiempo_T3: Segundos invertidos para realizar la tarea 3.
- Tiempo_T4: Segundos invertidos para realizar la tarea 4.
- Tiempo_Total: Segundos invertidos para realizar las cuatro tareas.

El primer paso antes de aplicar el ANOVA es comprobar si se cumplen los requisitos de normalidad, independencia e igualdad de varianzas. Para comprobar la normalidad, se aplica el estadístico K-S. La Figura 12.31 muestra el resultado de dicho estadístico. De acuerdo a la Tabla 12.18, se puede afirmar que se cumple la hipótesis nula que afirma que **los datos siguen una distribución normal**.

Prueba de Kolmogorov-Smirnov para una muestra

		Tiempo_T1	Tiempo_T2	Tiempo_T3	Tiempo_T4	Tiempo_Total
N		66	66	66	66	66
Parámetros normales ^a	Media	201,91	243,27	226,08	157,33	828,59
	Desviación típica	99,091	142,698	153,796	105,641	375,393
Diferencias más extremas	Absoluta	,125	,118	,140	,193	,149
	Positiva	,125	,118	,140	,193	,149
	Negativa	-,104	-,061	-,118	-,162	-,081
Z de Kolmogorov-Smirnov		1,014	,955	1,137	1,572	1,212
Sig. asintót. (bilateral)		,255	,321	,150	,014	,106

a. La distribución de contraste es la Normal.

Figura 12.31 Prueba K-S para Tiempo_T1, Tiempo_T2, Tiempo_T3, Tiempo_T4, Tiempo_Total

La Tabla 12.20 muestra los resultados del estadístico Durbin-Watson para comprobar la independencia de las muestras. Los valores están dentro de los márgenes para **considerar las muestras como independientes** (1,5 y 2,5).

Variable dependiente	Estadístico Durbin-Watson
Tiempo_T1	1,84
Tiempo_T2	2,05
Tiempo_T3	1,78
Tiempo_T4	1,71
Tiempo_Total	1,80

Tabla 12.20 Prueba de independencia de Durbin-Watson para la eficiencia

Por último, la Figura 12.32 muestra el resultado de la prueba de Levene para demostrar la homogeneidad de varianzas. En base al resultado de dicho test y a la Tabla 12.18, se puede concluir que se acepta la hipótesis nula que afirma la **existencia de homogeneidad de varianzas**.

Prueba de homogeneidad de varianzas

	Estadístico de Levene	gl1	gl2	Sig.
Tiempo_T1	,383	1	64	,538
Tiempo_T2	2,190	1	64	,144
Tiempo_T3	,267	1	64	,607
Tiempo_T4	,181	1	64	,672
Tiempo_Total	,113	1	64	,738

Figura 12.32 Prueba de Levene para la eficiencia

Una vez demostrado que se cumplen las tres condiciones, se aplica el ANOVA a las cinco variables dependientes que representan la eficiencia del usuario. Existe una hipótesis nula por cada uno de los tiempos estudiados (cinco). Las hipótesis nulas afirman que las medias de las poblaciones estudiadas son iguales. Por otro lado, las hipótesis alternativas afirman que al menos una de las poblaciones difiere de las demás en cuanto a su valor esperado.

ANOVA

		Suma de cuadrados	gl	Media cuadrática	F	Sig.
Tiempo_T1	Inter-grupos	5893,364	1	5893,364	,596	,443
	Intra-grupos	632340,091	64	9880,314		
	Total	638233,455	65			
Tiempo_T2	Inter-grupos	116115,341	1	116115,341	6,155	,016
	Intra-grupos	1207461,750	64	18866,590		
	Total	1323577,091	65			
Tiempo_T3	Inter-grupos	31496,371	1	31496,371	1,339	,252
	Intra-grupos	1505954,250	64	23530,535		
	Total	1537450,621	65			
Tiempo_T4	Inter-grupos	13643,667	1	13643,667	1,227	,272
	Intra-grupos	711751,000	64	11121,109		
	Total	725394,667	65			
Tiempo_Total	Inter-grupos	506664,273	1	506664,273	3,747	,057
	Intra-grupos	8653117,682	64	135204,964		
	Total	9159781,955	65			

Figura 12.33 ANOVA de la eficiencia con el factor experiencia

La Figura 12.33 muestra el resultado del ANOVA tras aplicar el factor experiencia en el uso de aplicaciones OlivaNOVA a las cinco variables dependientes. A partir del valor de la columna significancia y en base a la Tabla 12.18, se puede afirmar que la hipótesis nula de que las medias de las poblaciones estudiadas son iguales se puede aceptar. Es decir, **no**

existe una relación entre la eficiencia del usuario y la experiencia previa en el uso de aplicaciones OlivaNOVA.

La Figura 12.34 muestra el resultado del ANOVA tras aplicar el factor formas de uso a las cinco variables dependientes. A partir del valor de la columna significancia y en base a la Tabla 12.18, se puede afirmar que la hipótesis nula de que las medias de las poblaciones estudiadas son iguales se puede aceptar. Es decir, **no existe una relación entre la eficiencia del usuario y la incorporación o no de las formas de uso en las aplicaciones estudiadas.**

ANOVA

		Suma de cuadrados	gl	Media cuadrática	F	Sig.
Tiempo_T1	Inter-grupos	23221,879	1	23221,879	2,417	,125
	Intra-grupos	615011,576	64	9609,556		
	Total	638233,455	65			
Tiempo_T2	Inter-grupos	62930,970	1	62930,970	3,195	,079
	Intra-grupos	1260646,121	64	19697,596		
	Total	1323577,091	65			
Tiempo_T3	Inter-grupos	3136,742	1	3136,742	,131	,719
	Intra-grupos	1534313,879	64	23973,654		
	Total	1537450,621	65			
Tiempo_T4	Inter-grupos	858,242	1	858,242	,076	,784
	Intra-grupos	724536,424	64	11320,882		
	Total	725394,667	65			
Tiempo_Total	Inter-grupos	33773,470	1	33773,470	,237	,628
	Intra-grupos	9126008,485	64	142593,883		
	Total	9159781,955	65			

Figura 12.34 ANOVA de la eficiencia con el factor formas de uso

Por último, se estudia la interacción de los dos factores (experiencia y formas de uso). Esto nos permite detectar si dicha interacción afecta o no a la eficiencia del usuario. El resultado de este ANOVA se puede observar en la Figura 12.35. Al comparar los resultados de la columna significación con la Tabla 12.18, se puede concluir que se acepta la hipótesis nula, es decir, que **no hay relación entre la interacción de los factores experiencia y usabilidad con la satisfacción del usuario.**

Pruebas de los efectos inter-sujetos

Fuente	Variable dependiente	Suma de cuadrados tipo III	gl	Media cuadrática	F	Significación
Exp * Usab	Tiempo_T1	26096,485	1	26096,485	2,775	,101
	Tiempo_T2	6314,917	1	6314,917	,344	,560
	Tiempo_T3	714,008	1	714,008	,029	,864
	Tiempo_T4	6000,758	1	6000,758	,528	,470
	Tiempo_Total	34693,939	1	34693,939	,251	,618

Figura 12.35 ANOVA de la eficiencia estudiando dos factores: experiencia y formas de uso

12.6.2.3 Gráficos box and whiskers

Por último, se muestran los gráficos box and whiskers para las variables dependientes estudiadas en la eficiencia. Los gráficos de Tiempo_T1, Tiempo_T2, Tiempo_T3, Tiempo_T4 y Tiempo_Total son muy similares entre sí, por lo que sólo se van a mostrar los de Tiempo_Total a modo de ejemplo.

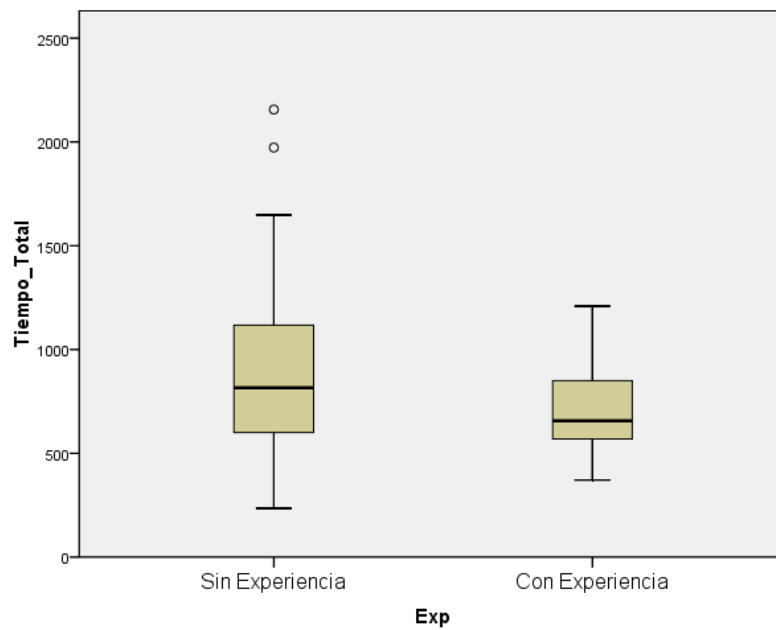


Figura 12.36 Gráfico box and whiskers de Tiempo_Total con el factor experiencia

La Figura 12.36 muestra el gráfico de la variable Tiempo_Total con el factor experiencia. Se puede apreciar cómo las medianas (línea dentro de cada caja) son similares tanto para los usuarios con experiencia en OlivaNOVA como para los que no la tienen. En ambos casos la distribución es asimétrica positiva. También se observa que el rango de valores para la mayoría de los usuarios sin experiencia es un poco mayor que para los usuarios con experiencia. Eso quiere decir que los usuarios sin experiencia tardan generalmente un poco más de tiempo en realizar las tareas.

La Figura 12.37 muestra el gráfico del estudio de la variable Tiempo_Total con el factor formas de uso. Las medianas son idénticas entre sí. La única diferencia es que para los usuarios que interactúan con formas de uso, la mediana es mucho más asimétrica positiva. Además, la mayoría de los valores tanto para los usuarios que interactúan con formas de uso como para los que interactúan sin ellas se mueven dentro de un rango de valores muy similar.

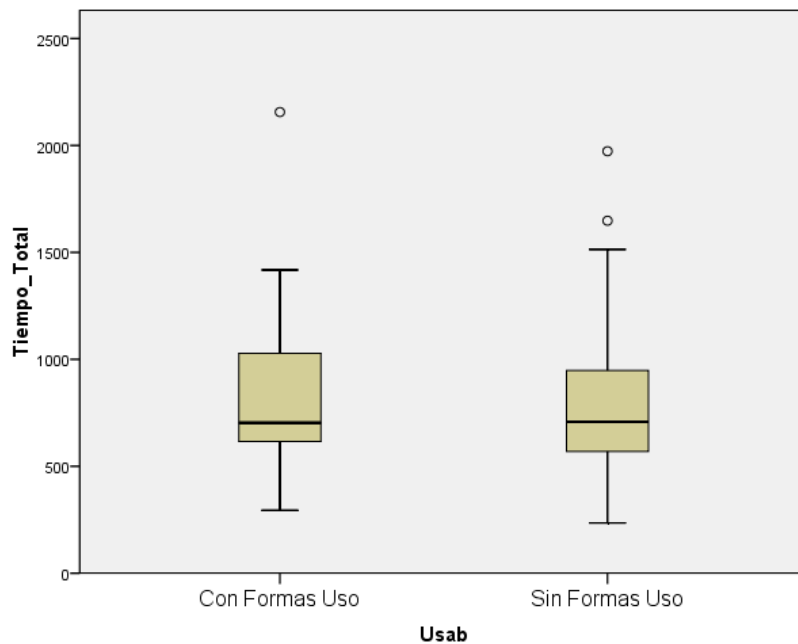


Figura 12.37 Gráfico box and whiskers de Tiempo_Total con el factor formas de uso

12.6.2.4 Conclusiones del análisis de la eficiencia

En base a los resultados obtenidos comparando las medias, el ANOVA y los gráficos box and whiskers, se puede afirmar que la hipótesis nula **H₂₀ (La eficiencia de los usuarios que usan aplicaciones Web con formas de uso es igual a la eficiencia de los usuarios que usan aplicaciones Web sin formas de uso) se cumple**. Por lo tanto, las formas de uso estudiadas en el experimento no contribuyen a mejorar la eficiencia del usuario, sino sólo su satisfacción. Es importante destacar que el hecho de que el usuario haga las tareas en menos tiempo no quiere decir que las haya finalizado correctamente. Varias formas de uso ralentizan la labor del usuario pero le previenen de errores, lo cual es normalmente preferible.

12.6.3 Conclusiones de la evaluación experimental

El objetivo principal del experimento era el de demostrar que la utilización de formas de uso en los sistemas mejora su usabilidad. Para ello se ha contado con un total de 66 participantes, número suficiente para extrapolar los resultados aquí obtenidos a otros usuarios de características similares a los estudiados, es decir, personas que hayan utilizado alguna vez aplicaciones Web. El hecho de que el experimento se haya hecho vía Web ha facilitado su difusión y que el número de participantes haya sido el suficiente.

A partir del análisis de resultados se puede concluir que **la satisfacción del usuario sí que mejora tras incorporar formas de uso, especialmente si se trata de usuarios novatos**. En cambio, **la eficiencia de los usuarios se mantiene idéntica tras la incorporación de formas de uso en el sistema**, o incluso empeora ligeramente si los usuarios son expertos. Al aumentar la satisfacción del usuario se mejora la usabilidad [ISO 9241-11, 1998], por lo que queda demostrado que la usabilidad de los sistemas mejoras tras la incorporación de formas de uso.

Las formas de uso que están relacionadas con la mejora de usabilidad son las siguientes:

- Mensaje de aviso (WU_W1)

- Especificar el tipo de visualización del campo de entrada (WU_STE1)
- Definición de máscaras (WU_STE2)
- Valores por defecto (WU_STE3)
- Ayuda dinámica (WU_MH1)

En cambio, no existe relación para las siguientes formas de uso:

- Informar del éxito o fracaso en la ejecución del servicio (WU_SSF1)
- Mostrar el estado de las acciones (WU_SSF3)
- La usabilidad general del sistema

Si las formas de uso se incorporaran en un método de desarrollo TTM, el analista podría fácilmente añadir o quitar las distintas formas de uso en el sistema dependiendo del tipo de usuarios (novatos o expertos) que interactúe más frecuentemente con el sistema. Si el sistema es utilizado por muchos usuarios con distintos niveles de conocimiento de la aplicación, entonces el analista debería utilizar el mayor número posible de formas de uso. En cambio, si el sistema es utilizado la mayor parte del tiempo por los mismos usuarios, el analista no debería utilizar muchas formas de uso a fin de evitar perder eficiencia.

En el caso de OlivaNOVA, un mismo sistema se podría modelar con distintas vistas. Una vista representaría el sistema con formas de uso especialmente diseñado para novatos en el ámbito de la aplicación y otra vista con pocas formas de uso representaría el sistema especialmente diseñado para expertos. El uso de una u otra vista dependería del tipo de usuario que se registrara en el sistema para interactuar con él.

Capítulo 13

Conclusiones y Trabajos Futuros

En este último capítulo se presentan las conclusiones extraídas de esta investigación y las líneas de investigación futuras en el marco de este trabajo. El objetivo principal de la tesis es **obtener un método para incorporar mecanismos de usabilidad existentes en la literatura dentro de un método de desarrollo MDD**. Más concretamente, la propuesta está especialmente pensada para las Tecnologías de Transformación de Modelos, que son métodos MDD en los cuales la transformación entre el espacio del problema (modelos conceptuales) y el espacio de la solución (código en un lenguaje de programación) se hace, gracias a un compilador de modelos, lo más automáticamente posible. El método propuesto se ha llamado MIMAT. De entre todos los tipos de sistemas existentes, **esta tesis se enmarca dentro del desarrollo de sistemas de gestión de la información, abarcando tanto aplicaciones Web como de Escritorio**.

El principal problema del analista que intenta incorporar usabilidad a los sistemas desarrollados, es el alto coste de tiempo que supone el tratar con la usabilidad durante el proceso de desarrollo. La incorporación de los mecanismos de usabilidad dentro de un desarrollo de sistemas dirigido por modelos acorta estos tiempos, ya que la usabilidad se puede incorporar fácilmente a través de primitivas conceptuales que posteriormente son traducidas a código. El analista podría dedicar todo su esfuerzo a la construcción de los modelos conceptuales, independientemente de la tecnología sobre la cual se implemente posteriormente el sistema.

MIMAT está pensado para que lo aplique el diseñador de la TTM, ya que deberá incorporar nuevos cambios con el fin de dar soporte a los mecanismos de usabilidad. Una vez dichos mecanismos se han incorporado a la TTM, el analista que trabaja con dicha TTM podrá utilizar nuevas primitivas conceptuales para trabajar con los mecanismos. Por tanto, el rol del diseñador de la TTM y del analista, son distintos. El primero trabaja desarrollando y mejorando la TTM, mientras que el segundo trabaja con la TTM con el fin de construir sistemas software. El esfuerzo dedicado a pensar en las primitivas necesarias para representar los mecanismos de usabilidad y la estrategia de generación de código a partir de dichas primitivas, sólo se tiene que hacer una vez por el diseñador de la TTM. Una vez incorporados estos cambios a la TTM, el analista que trabaja con la TTM realiza poco esfuerzo para desarrollar aplicaciones usables.

Esta investigación ha revisado las distintas aproximaciones que se han realizado tanto en el ámbito IPO como en el de la IS para incorporar la usabilidad desde las primeras fases del proceso de desarrollo software. Se han comparado las propuestas de la literatura que pretenden incorporar la usabilidad desde la fase de diseño de la arquitectura, las que pretenden incorporar la usabilidad a nivel de requisitos, las que pretenden incorporar la usabilidad utilizando patrones de diseño, y por último, las que pretenden incorporar la usabilidad en un entorno MDD. **De todas las propuestas encontradas, ninguna resuelve el problema de cómo incorporar usabilidad en MDD, de ahí la importancia y relevancia de esta tesis dentro del área.**

De todos los trabajos existentes en la literatura, esta tesis se ha basado en los mecanismos de usabilidad definidos por Juristo. Estos mecanismos fueron definidos para incorporarlos en un proceso de desarrollo tradicional, es decir, desarrollo totalmente de forma manual. La elección de estos mecanismos ha estado motivada por la existencia de dos elementos en su definición que son utilizados en el método MIMAT: las guías de captura de requisitos y los patrones de diseño. Las guías de captura de requisitos se han utilizado para extraer qué propiedades de los mecanismos se deben representar en los modelos conceptuales. Por otro lado, los patrones de diseño se han utilizado para definir las reglas de transformación de modelo a código utilizadas para implementar la funcionalidad de los mecanismos de usabilidad.

El método MIMAT creado a partir de la definición de los mecanismos de usabilidad **consta de cuatro pasos: definición de formas de uso, identificación de propiedades, definición de primitivas conceptuales, identificación de los cambios necesarios en el compilador de modelos.** De estos cuatro pasos, los dos primeros son válidos para cualquier TTM. En cambio, los dos últimos dependen de la TTM sobre la cual se aplique MIMAT, ya que cada TTM tiene un modelo conceptual y compilador de modelos propios. Es importante remarcar que el método MIMAT será aplicado una sola vez por el diseñador de la TTM. Una vez la TTM haya incorporado los mecanismos de usabilidad, el analista que trabaje con esa TTM para el desarrollo de sistemas, puede utilizar las nuevas primitivas conceptuales que representan aspectos de usabilidad.

Como ejemplo de TTM para aplicar MIMAT se ha elegido OO-Method, ya que dispone de una herramienta industrial que la implementa (llamada OlivaNOVA). Esta herramienta tiene como ventaja que proporciona un buen contexto para desarrollar experimentos. Además, el modelo conceptual de OO-Method es lo suficientemente abstracto como para incorporar nuevas primitivas con semántica relacionada con la usabilidad. Para cada uno de los mecanismos de usabilidad definidos por Juristo, **se han extraído una serie de propiedades. Estas propiedades provocan cambios en los modelos conceptuales y en el compilador de modelos de OO-Method.** Estos cambios se han detallado a lo largo de varios capítulos de la tesis. **Los resultados obtenidos al aplicar MIMAT a OO-Method sirven de ejemplo para poder aplicar MIMAT a otras TTMs.**

Otra de las contribuciones de la tesis consiste en justificar que el método MIMAT se puede aplicar a TTMs reales y que dicho esfuerzo mejora la usabilidad de las aplicaciones generadas. Por un lado, para justificar que MIMAT tiene viabilidad, se ha mostrado cómo se aplicaría a OO-Method. Para ello, se han creado prototipos de ventanas en las cuales el analista de OO-Method podría trabajar con las propiedades de usabilidad de forma abstracta representados mediante primitivas conceptuales. **Estos prototipos ilustran también cómo se podrían añadir las propiedades de usabilidad a otras TTMs distintas de OO-Method.**

La propuesta de esta investigación se ha evaluado experimentalmente. La TTM elegida para este experimento ha sido nuevamente OO-Method y su

herramienta industrial OlivaNOVA. En el experimento han participado tanto usuarios que han interactuado previamente con aplicaciones generadas por OlivaNOVA como usuarios sin experiencia previa en este tipo de aplicaciones. En cada uno de estos grupos ha habido usuarios que han interactuado con una aplicación que incorporaba mecanismos de usabilidad y otros usuarios han interactuado con la misma aplicación pero sin incorporar dichos mecanismos. De esta manera, se ha podido comparar si la usabilidad percibida por el grupo que interactuó con la aplicación que usaba mecanismos de usabilidad era mejor o no con respecto al grupo que interactuó con la aplicación que no usaba mecanismos. **Los resultados demuestran que la satisfacción de los usuarios que interactuaron con la aplicación que usaba mecanismos era mayor.** La mejora en la satisfacción implica una mejora en la usabilidad del sistema [ISO 9241-11, 1998]. Además, también se ha demostrado que la eficiencia del usuario (tiempo en realizar sus tareas) no varía en base al uso de los mecanismos de usabilidad. Por lo tanto, **el esfuerzo que supone aplicar el método MIMAT a las TTMs está justificado por los beneficios en la satisfacción,** que repercute en beneficios de la usabilidad.

Son varios los trabajos futuros que se pueden realizar. Tras la incorporación de los mecanismos de usabilidad a OO-Method, un primer trabajo es la definición de una guía de buenas prácticas para que el analista genere aplicaciones lo más usable posible. En esta tesis se ha mostrado cómo el analista de OO-Method puede representar cada una de las propiedades de usabilidad. Sin embargo, estas guías no son suficientes para asegurar que las aplicaciones generadas sean usables. Depende del criterio del analista el utilizar una primitiva de modelado u otra. Por tanto, como trabajo futuro, es interesante definir una guía de buenas prácticas que contemplen la mejor combinación de las primitivas conceptuales para obtener el mejor valor de usabilidad posible del producto software desarrollado.

Por otro lado, al estar la usabilidad embebida dentro de modelos conceptuales, debe ser posible medirla a partir de dichos modelos sin necesidad de que el usuario interactúe con dicha aplicación. Como trabajo futuro, se pretende definir una serie de métricas basadas en los modelos conceptuales de manera que proporcionen una estimación sobre cuánto de usable es la aplicación antes de generar su código. Esto permitirá al analista

anticiparse a posibles carencias de usabilidad antes de realizar las pruebas de usabilidad más exhaustivas con usuarios.

Otra de las líneas futuras consiste en la extensión del método MIMAT con mecanismos de usabilidad diferentes a los utilizados en la tesis. Existen otros mecanismos de usabilidad en la literatura que podrían ser interesantes también para el tipo de aplicaciones abordadas en esta investigación (aplicaciones de gestión de los sistemas de información). Además, también existen mecanismos de usabilidad especializados para otros tipos de sistemas, como por ejemplo, multimedia. Convendría hacer un estudio exhaustivo para ver los cambios que se deberían incorporar a MIMAT con el fin de dar soporte a otros mecanismos.

Como último trabajo futuro, convendría hacer más estudios empíricos con el fin de valorar el grado de mejora en la satisfacción que provoca cada uno de los mecanismos de usabilidad. En esta investigación sólo se han estudiado unos pocos, pero convendría hacer un estudio con todos los mecanismos. Esto junto con los estudios ya existentes en la literatura que miden cuánto de costoso es la implantación de cada uno de los mecanismos, daría una visión de cuáles son más sencillos de incorporar y cuáles mejoran más la usabilidad, es decir, cuál obtiene un mejor ratio beneficio-coste.

Para terminar, resaltar la importancia que poco a poco va adquiriendo la usabilidad dentro de los métodos MDD con el fin de obtener sistemas software de calidad. Este trabajo va encaminado en esa dirección.

Referencias

- [Abrahao, 2005] Abrahão, S., Pastor, Ó. and Olsina, L. (2005). A Quality Model for Early Usability Evaluation. INTERACT05 International COST 294 Workshop on User Interface Quality Models (UIQM), Rome, Italy. pp: 68 – 77
- [Abrahao, 2006] Abrahao, S., Insfrán, E. (2006). Early Usability Evaluation in Model Driven Architecture Environments. Sixth International Conference on Quality Software (QSIC'06), Beijing, China. pp. 287-294
- [Adikari, 2006] Adikari, S. and C. McDonald (2006). User and Usability Modeling for HCI/HMI: A Research Design. Information and Automation, 2006. ICIA 2006. pp. 151-154
- [Alexander, 1977] Alexander, C., Ishikawa, S. & Silverstein, M. (1977). A Pattern Language: Towns, Buildings, Construction, Oxford University Press.
- [Aquino, 2008] Aquino, N., Vanderdonckt, J., Valverde, F., Pastor, O. (2008). Using Profiles to Support Model Transformations in the Model-Driven Development of User Interfaces. Proc. of 7th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2008, Albacete, Spain, Springer, Berlin.
- [Barfield, 1993] Barfield, L. (1993). "The User Interface. Concepts & Design." EEUU, Addison-Wesley.
- [Basili, 88] Basili, V. and Rombach, H. (1988). "The TAME Project: Towards Improvement-Oriented Software Environments." IEEE Transactions on Software Engineering ,14, pp: 758-773.
- [Bass, 2003] Bass, L. and J. Bonnie (2003). "Linking usability to software architecture patterns through general scenarios." The journal of systems and software **66**: 187-197.
- [Bastien, 1993] Bastien, J. M., Scapin, D. (1993). "Ergonomic Criteria for the Evaluation of Human-Computer Interfaces." Rapport technique de l'INRIA: 79.
- [Benson, 2002] Benson, C., A. Elman, Nickell, S., Robertson, C.. (2002). GNOME Human Interface Guidelines (1.0).
<http://developer.gnome.org/projects/gup/hig/1.0/index.html>. Última visita: junio 2008
- [Bevan, 2000] Bevan, N. and I. Bogomolni (2000). Incorporating User Quality Requirements in the Software Development Process. International Software and Internet Quality Week Conference (QWE).

- [Borchers, 2000] Borchers, J. (2000). A pattern approach to interaction design. International Conference on Design Interactive Systems, New York, ACM Press.pp. 369 - 378
- [BORE, 2008] BORE: <http://cse-ferg41.unl.edu/bore.html> Última visita: junio 2008
- [CARE, 2008] CARE Technologies S.A. www.care-t.com . Última visita: julio 2008.
- [Chung, 1999] Chung, L., B. A. Nixon, et al. (1999). Non-Functional Requirements in Software Engineering. Boston, Kluwer Academic Publishers.
- [Comstock, 1996] Comstock, E. and W. Duane (1996). Embed User Values in System Architecture: The Declaration of System Usability. CHI 96.
- [Coram, 1996] Coram, T., Lee, L. (1996). Experiences: A Pattern Language for User Interface Design. <http://www.maplefish.com/todd/papers/experiences/Experiences.html> Última visita: junio 2008
- [Cortex, 2009] IT Cortex Project Failure Statistics http://www.it-cortex.com/Stat_Failure_Cause.htm. Última visita: abril 2009
- [Cysneiros, 2003] Cysneiros, L. M. and A. Kushniruk (2003). Bringing Usability to the Early Stages of Software Development. International Requirements Engineering Conference, IEEE.pp. 359- 360
- [Daniels, 2007] Daniels, C. B., LaMarsh W.J. (2007). Complexity as a Cause of Failure in Information Technology. Project Management. IEEE Intl Conf on System of Systems Engineering, (SoSE '07).
- [De la Vara, 2008] De la Vara, J. L., Sánchez, J. and Pastor, O. (2008). Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. CAISE 2008, Springer LNCS 5074. pp: 213-227.
- [Díaz, 2003] Díaz I, Losavio F, Matteo A , Pastor O (2003). A Specification Pattern for Use Cases. Information & Management Journal, (Elsevier Science B.V.) (0378-7206).
- [España, 2006] España, S., Pederiva, I., Panach, I., Abrahão, S., Pastor, O. (2006). Evaluación de la Usabilidad en un Entorno de Arquitecturas Orientadas a Modelos. IDEAS 2006, Argentina. pp: 331-344
- [España, 2009] España, S., Arturo, G.-d.-R. R. and Pastor, Ó. (2009). Communication Analysis: a Requirements Engineering method for information

systems. 1st International Conference on Advanced Information Systems (CAiSE'09). Amsterdam, The Netherlands., Springer LNCS 5565. pp: 530-545.

- [Estrada, 2006] Estrada, H., Martínez, A., Pastor, Ó. and Mylopoulos, J. (2006). An empirical evaluation of the i* framework in a model-based software generation environment. CAISE 2006, Lecture Notes in Computer Science 4001, Springer Verlag. pp. 513-527.
- [Folmer, 2004] Folmer E., Bosch J. (2004) Architecting for usability: A Survey, Journal of Systems and Software, 70(1) 61-78.
- [Goguen, 1993] Goguen, J. A. and C. Linde (1993). Techniques for Requirements Elicitation. Requirements Engineering. pp. 152-164
- [Grau, 2006] Grau, G., Cares, C., Franch, X., Navarrete, F. (2006). A comparative analysis of i*-based agent-oriented modelling techniques. Eighteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2006), San Francisco, CA, USA. pp. 657-663.
- [Griffiths, 2002] Griffiths, R. (2002). The Brighton Usability Pattern Collection, <http://www.cmis.brighton.ac.uk/research/patterns/home.html>.
- [Hailpern, 2006] Hailpern, B., Tarr, P. (2006). "Model-Driven Development: the Good, the Bad, and the Ugly." IBM Syst. J. 45(3): 451-461.
- [Henninger, 2005] Henninger, S. and P. Ashokkumar (2005). An Ontology-Based Infrastructure for Usability Design Patterns. Semantic Web Enabled Software Engineering (SWESE), Galway, Ireland. pp. 41-55
- [IBM, 2005] IBM, 2005. Cost Justifying Ease of Use. Available at: http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/23.
- [Insfrán, 2002] Insfrán, E., Pastor, O., Wieringa, R. (2002). Requirements Engineering-Based Conceptual Modelling. Requirements Engineering, Vol. 7, Issue 2, p. 61-72. Springer-Verlag.
- [ISO 13407, 2001] ISO 13407: User centred design process for Interactive systems.
- [ISO 9241-11, 1998] ISO 9241-11 (1998) Ergonomic requirements for office work with visual display terminals - Part 11: Guidance on Usability.
- [ISO TR 18529, 2000] ISO TR 18529: Human-centred lifecycle process descriptions.

- [ISO/IEC 9126-1, 2001] ISO/IEC 9126-1 (2001), Software engineering - Product quality - 1: Quality model.
- [Juristo, 2003] Juristo, N., Moreno, A., Sánchez, M. (2003) Techniques and Patterns for Architecture-Level Usability Improvements. Deliv. 3.4 STATUS project.
<http://www.ls.fi.upm.es/status>
- [Juristo, 2007, a] Juristo, N., A. M. Moreno, et al. (2007). Analysing the impact of usability on software design. *Journal of Systems and Software*. **80**: 1506-1516.
- [Juristo, 2007, b] Juristo, N., A. M. Moreno, et al. (2007). Guidelines for Eliciting Usability Functionalities. *IEEE Transactions on Software Engineering*. **33**: 744-758.
- [Kozima, 2005] Kozima, A., T. Kiguchi, et al. (2005). "A System To Guide Interview-Driven Requirements Elicitation Work: Domain—Specific Navigation Using The Transition Pattern Of Topics." *Journal of Integrated Design & Process Science* **9**(4): 27-39.
- [Krasner, 1998] Krasner, G. E. and S. T. Pope (1998). "A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80." *Journal Of Object Oriented Programming (JOOP)*.
- [Lauesen, 1998] Lauesen, S. (1998). Usability Requirements in a Tender Process. *Computer Human Interaction Conference, 1998, Australia*.pp. 114-121
- [Lawrence, 2001] Lawrence, B., K. Wiegers, et al. (2001). The top risk of requirements engineering. *IEEE Software*. **18**: 62-63.
- [Leite, 1996] Leite, J. C. S. d. P. and Gilvaz, A. P. P. (1996). Requirements Elicitation Driven by Interviews: The Use of Viewpoints. *Proceedings of the 8th International Workshop on Software Specification and Design (IWSSD '96)*.
- [MDA, 2008] MDA Guide V1.0.1: <http://www.omg.org/docs/omg/03-06-01.pdf>,
Última visita: julio 2008.
- [Mellor, 2002] Mellor, S., K. Scott, Uhl, A., Weise, D. (2002). *Model-Driven Architecture*, Springer Berlin / Heidelberg: 233-239.
- [Mellor, 2003] Mellor, S. J., Clark, A. N., Futagami, T. (2003). Guest Editors' Introduction: Model-Driven Development. *IEEE Software*. 20: 14-18.
- [Miller, 2001] Miller, J. and J. Mukerji (2001). *Model Driven Architecture (MDA)*. A draft with annotations of issues to resolve, Architecture Board ORMSC.

-
- [MOF QVT, 2008] MOF QVT: [http:// www.omg.org/cgi-bin/apps/doc?ptc/05-11-01](http://www.omg.org/cgi-bin/apps/doc?ptc/05-11-01), Última visita: julio 2008.
- [Molina, 2003] Molina, P. J. (2003). Especificación de interfaz de usuario: de los requisitos a la generación automática. PhD. Departamento de Sistemas Informáticos y Computación. Valencia, Universidad Politécnica de Valencia: 382.
- [Nielsen, 1993] Nielsen, J. (1993). Usability Engineering, AP Professional.
- [Olivé, 2007] Olivé, A. (2007). Conceptual Modeling of Information Systems, Springer.
- [Panach, 2007, a] Panach, I., Condori, N., Valverde, F., Aquino, N., Pastor, O. (2007). Towards an Early Usability Evaluation for Web Applications. MENSURA 2007, LNCS 4895, pp.32-45
- [Panach, 2007, b] Panach, I., España, S., Pederiva, I., Pastor, O. (2007). Capturing Interaction Requirements in a Model Transformation Technology Based on MDA. Journal of Universal Computer Science (JUCS).
- [Panach, 2008] Panach, I., Condori, N., Valverde, F., Aquino, N., Pastor, O. (2008). Understandability Measurement in an Early Usability Evaluation for Model-Driven Development: An Empirical Study. Empirical Software Engineering and Measurement (ESEM), Kaiserslautern, Germany., ACM-IEEE, pp. 354-356
- [Pastor, 2007] Pastor, O., Molina, J. (2007). Model-Driven Architecture in Practice. Valencia, Springer.
- [Paternò, 2004] Paternò, F. (2004). ConcurTaskTrees: An Engineered Notation for Task Models. The Handbook of Task Analysis for Human-Computer Interaction. D. Diaper, N. Stanton and N. A. Stanton. London, United Kingdom, Lawrence Erlbaum Associates: 483-501.
- [Perzel, 1999] Perzel, K., Kane, D. (1999). Usability Patterns for Applications on the World Wide Web. PloP'99 Conference.
- [Pointer, 2008] Pointer (2008). Patterns of Interaction: a Pattern Language for CSCW.
<http://www.comp.lancs.ac.uk/computing/research/cseg/projects/pointer/pointer.html>. Última visita: agosto 2008
- [Potts, 1993] Potts, C. (1993). Software-Engineering Research Revisited. IEEE Software. **10**: 19-28.

- [Selic, 2003] Selic, B. (2003). "The Pragmatics of Model-Driven Development." IEEE software **20**(5): 19-25.
- [Sendall, 2003] Sendall, S. and W. Kozaczynski (2003). "Model Transformation: The Heart and Soul of Model-Driven Software Development." IEEE Software **20**(5): 42-45.
- [Shackel, 1991] Shackel, B. (1991). Usability - context, framework, design and evaluation, Human Factors for Informatics Usability. Cambridge University Press, Cambridge, 21-38.
- [STATUS, 2008] Proyecto STATUS: <http://is.ls.fi.upm.es/status>. Última visita: septiembre-2008
- [Sutcliffe, 99] Sutcliffe, A. and Galliers, J. (1999). Human Errors and System Requirements. Proceedings, RE'99 The Fourth International Symposium on Requirements Engineering. IEEE.
- [Tao, 2005] Tao, Y. (2005). Work in Progress - Introducing Usability Concepts in Early Phases of Software Development. Frontiers in Education, 2005. FIE '05. Proceedings 35th Annual Conference. pp. T4C- 7-8
- [Thibodeau, 2002] Thibodeau, P. (2002). Users Begin to Demand Software Usability Tests., ComputerWorld. Disponible en : <http://www.computerworld.com/softwaretopics/software/story/0,10801,76154,00.html>.
- [Tidwell, 2005] Tidwell, J. (2005). Designing Interfaces, O'Reilly Media.
- [UML, 2008] UML: <http://www.uml.org/> Última visita: mayo 2008
- [Welie, 2000] Welie, M. v. and H. Traetteberg (2000). Interaction Patterns in User Interfaces. 7th. Pattern Languages of Programs Conference, Illinois, USA.
- [Wohlin, 99] Wohlin, C., Runeson, P. and Höst, M. (1999). Experimentation in Software Engineering: An Introduction, Springer.
- [Yu, 1997] Yu, E. (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. IEEE Int. Symp. on Requirements Engineering. pp. 226-235
- [Visauta, 1997] Visauta, B. (1997). Análisis estadístico con SPSS para Windows : Estadística básica, Vol 1. McGraw-Hill/Interamericana de España.

[Visauta, 1998] Visauta, B. (1998). Análisis estadístico con SPSS para Windows : estadística multivariante, Vol 2. McGraw-Hill/Interamericana de España.

Anexo I: Mecanismos de Usabilidad

Este anexo muestra la descripción de todos los mecanismos de usabilidad utilizados en la tesis tal y como aparecen publicados en [Juristo, 2007, a] y [Juristo, 2007, b]

1. Feedback

1.1 System Status Feedback

IDENTIFICADOR	
Nombre: System Status Feedback	
Familia: Feedback	
Alias: Status Display Modelling Feedback Area	
PROBLEMA	
Qué información se debe obtener y especificar para que la aplicación proporcione a los usuarios retroalimentación del estado del sistema.	
CONTEXTO	
Cuando cambios que son importantes para el usuario ocurren durante: <ul style="list-style-type: none"> • La ejecución de una tarea • Porque no hay suficientes recursos en el sistema • Porque recursos externos no funcionan correctamente Ejemplos de este mecanismo se pueden encontrar en la barra de estado de las aplicaciones Windows o en sistemas que muestran horarios de trenes y aviones	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Los expertos IPO sostienen que el usuario quiere ser informado cuando ocurre un cambio de estado.	Los cambios en el estado del sistema pueden ser provocados por peticiones del usuario, por otras tareas, o cuando haya un

	<p>problema con un recurso externo u otro recurso del sistema.</p> <p>1.1 ¿El usuario quiere que el sistema le avise de los estados del sistema?, en ese caso, ¿de cuáles?</p> <p>1.2 ¿El usuario quiere que el sistema le avise de fallos del sistema (cualquier operación que el sistema no es capaz de completar, pero no son fallos provocados por entradas incorrectas del usuario)? En ese caso ¿Cuáles?</p> <p>1.3 ¿El usuario quiere que el sistema avise si no hay suficientes recursos para ejecutar las órdenes en curso? En ese caso, ¿qué recursos?</p> <p>1.4 ¿El usuario quiere que el sistema le avise si hay problemas con los recursos externos o dispositivos con los que el sistema interactúa? En ese caso, ¿Cuáles?</p>
<p>2. Se deben elegir ventanas bien diseñadas para la información a mostrar. Éstas deben ser no bloqueante si la información no es importante, pero bloqueante si algo importante ocurre. Las ventanas se deben colocar de manera que enfatizen las cosas importantes, den menos importancia a las triviales y prevengan confusiones entre distintas partes de la información mostrada. La atención se debe centrar en la información importante con colores brillantes, parpadeo, sonido o los 3.</p>	<p>2.1 ¿Qué información se debe mostrar al usuario?</p> <p>2.2 ¿Qué información se tendrá que mostrar de manera bloqueante porque pertenece a una situación crítica? Esta información se suele representar en una ventana emergente de la ventana principal que impide al usuario seguir hasta que no la cierre.</p> <p>2.3 ¿Cuál de esta información tendrá que ser resaltada porque es importante pero no crítica? Usando diferentes colores, sonidos, tamaños, etc.</p> <p>2.4 ¿Cuál de esta información será mostrada en el área del estado?</p> <p>Para cada componente que muestre el estado del sistema, de acuerdo a su</p>

	<p>importancia, las posibilidades de visualización va desde los componentes bloqueantes (por ejemplo, una ventana en el área principal que impide al usuario continuar hasta que sean cerradas), pasando por resaltar ciertas zonas (con diferentes colores, sonidos, movimientos o tamaños) hasta las percepciones del ojo humano (como la identificación de un icono ubicado en el área del estado del sistema). Se puede observar que durante el proceso de elicitación de requisitos, el debate sobre el tipo exacto de respuesta se puede dejar hasta la fase de diseño de la interfaz. Es en el diseño cuando se debe pensar sobre la importancia de la información a mostrar y el tipo de ventana (bloqueante, resaltada o estándar).</p>
<p>3. Respecto a la localización del indicador de retroalimentación, la literatura IPO menciona que los usuarios prefieren un lugar donde sepan que pueden encontrar fácilmente su información sobre el estado. A parte del espacio donde los usuarios trabajan en la ventana, ellos prefieren ver la retroalimentación en el centro o en la parte superior de la ventana, y es menos agradable para ellos que se encuentre al pie de la ventana. La práctica habitual de colocar la información sobre cambios del estado en una línea que se muestra en el pie de la ventana es desaconsejable, especialmente si tiene un trazo fino sobre un fondo gris. Se debe recordar que las personas nacidas en la cultura</p>	<p>3.1 ¿Las personas que usan el sistema son de diferentes culturas? En ese caso, el sistema debe presentar el estado del sistema en la forma adecuada (de acuerdo a la cultura del usuario). Así que se debe preguntar al usuario sus costumbres y cultura.</p> <p>3.2 ¿Cuál es el mejor lugar para ubicar la información de retroalimentación de cada situación?</p>

<p> europea o americana están acostumbradas a leer de izquierda a derecha y de arriba abajo, y algo colocado en la esquina superior izquierda se verá fácilmente.</p>	
<p>Guía de Especificación de Mecanismos de Usabilidad:</p>	
<p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <ul style="list-style-type: none"> - El estado del sistema que se debería relatar es X. La información a mostrar en el área de estado es T. La información a resaltar es I. La información bloqueante es B. - El sistema software necesitará suministrar retroalimentación sobre los fallos I, II, III ocurridos en los servicios A, B, C respectivamente. La información relacionada con los fallos I, II, etc. Se debe mostrar en el área de estado L. La información relacionada con los fallos III, IV, etc. Se debe mostrar en formato resaltado. La información relacionada con los fallos V, VI, etc. Se debería mostrar en formato bloqueante. - El sistema software proporciona retroalimentación sobre los recursos D, E, F cuando falla IV, I y VI, respectivamente. La información a presentar sobre esos recursos es O, P, Q. La información que está relacionada con los fallos I, II se debe mostrar en el área de estado L. La información relacionada con los fallos III, IV se debe mostrar en formato resaltado. La información relacionada con los fallos V, VI se debe mostrar en formato bloqueante. - El sistema software necesitará suministrar retroalimentación a los recursos externos G, J, K, cuando los fallos VII, VIII y IX ocurran respectivamente. La información a presentar sobre esos recursos es R, S, T. La información relacionada con los fallos I, II se debe mostrar en el área de estado L. La información relacionada con los fallos III, IV se debe mostrar con formato resaltado. La información relacionada con los fallos V, VI se debe mostrar de manera bloqueante. 	

1.2 Interaction Feedback

<p>IDENTIFICADOR</p>
<p>Nombre: Interaction Feedback</p>
<p>Familia: Feedback</p>
<p>Alias: Interaction Feedback Modelling Feedback Area</p>
<p>PROBLEMA</p>

Qué información debe ser obtenida y especificada para mostrar al usuario que el sistema ha escuchado su petición.	
CONTEXTO	
Cuando el usuario realiza un evento de interacción, como un clic de ratón, movimiento de ratón, apretar una tecla, etc. El sistema debe informar al usuario de que la interacción se ha aceptado.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Ofrecer una retroalimentación proporcional a la escala del evento de interacción y su importancia, solo para confirmar que el sistema se ha percatado del evento [Griffiths, 2002]. Esta funcionalidad se puede presentar: marcando un botón, sombreando una palabra, cambiando la forma del cursor de los objetos involucrados, etc. Se debe proporcionar siempre una retroalimentación visual y permitir al usuario habilitar retroalimentaciones con sonido para todos los eventos de interacción.	1.1 Para cada acción que el usuario solicite al sistema se debe decidir, qué tipo de retroalimentación mostrará el sistema (marcando un botón, resaltando una palabra, cambiando el tamaño del cursor o los objetos involucrados) Se debe verificar que la aplicación proporciona la retroalimentación en 0.1 milisegundos después de cada vez que se pulse una tecla, se mueva el ratón, o el usuario haga cualquier otra actividad sobre el sistema.
Guía de Especificación de Mecanismos de Usabilidad:	
La siguiente información debe ser instanciada en el documento de requisitos: - El sistema ha de responder a los eventos de interacción A, B, C. C. La respuesta del sistema se hará en 0.1 milisegundos después de la interacción del usuario.	

1.3 Progress Feedback

IDENTIFICACIÓN
Nombre: Progress feedback
Familia: Feedback
Alias: Progress Indicator

Progress Show Computer is Thinking, Time to Do Something Else Modelling Feedback Area	
PROBLEMA	
Qué información debe ser obtenida y especificada para proporcionar a los usuarios la información que está relacionada con la evolución de las tareas solicitadas.	
CONTEXTO	
Cuando un proceso interrumpe la interfaz de usuario durante dos segundos o más.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Para servicios que se estén ejecutando durante 2 o más segundos: si el proceso es crítico, los usuarios no deberían poder hacer otra cosa hasta que este servicio se completara. Si el servicio no es crítico y dura más de 5 segundos, los usuarios deberían poder ejecutar otra acción si lo desean. Se debería avisar al usuario cuando los servicios que se estén ejecutando finalicen.	1.1 ¿Qué servicios es probable que duren más de 2 segundos y cuales de ellas son críticas? 1.2 ¿Cómo se informará al usuario cuando el proceso termine?
2. Mostrar un indicador animado que indique cuánto progreso se ha hecho. Además se debería informar al usuario textual o gráficamente de lo siguiente: <ul style="list-style-type: none"> • Qué se está ejecutando actualmente. • Qué proporción de la operación se ha realizado. • Cuánto tiempo falta para terminar. • Cómo parar la operación o cancelarla si el tiempo que falta para terminar es mayor de 10 segundos. Si el tiempo restante se puede calcular, se debe usar un indicador de tiempo de progreso o un indicador de la porción que falta por completarse. Si el tiempo no se puede estimar pero el proceso tiene fases	2.1. ¿Cómo se informará al usuario sobre el progreso de los diferentes servicios y qué información se desea para cada uno? 2.2. Se debe verificar que la aplicación no consume más de 1 minuto para mostrar el indicador de progreso y actualizar la retroalimentación a una velocidad que dé la impresión al usuario de que la operación aun se sigue realizando, por ejemplo cada 2 segundos.

<p>identificables, se deben indicar las fases completadas y las fases que restan. Se debe usar un checklist de progreso si no existe ninguna de estas posibilidades, para indicar el número de unidades procesadas. Si no se puede cuantificar cuánto tiempo llevará el proceso entonces simplemente se debería mostrar un indicador como que el proceso sigue en marcha, usando para ello un indicador de progreso indeterminado.</p>	
<p>Guía de Especificación de Mecanismos de Usabilidad:</p>	
<p>La siguiente información se debe instanciar en los documentos de requisitos:</p> <ul style="list-style-type: none"> - Las tareas U, V, Z necesitan más de 2 segundos, por lo tanto necesitarán una retroalimentación de progreso. Para ello, la información a mostrar será R, S, T en la parte A, B, C, respectivamente. - La información debe aparecer en menos de 1 segundo y se debe refrescar cada 2 segundos. 	

1.4 Warning

<p>IDENTIFICACION</p>	
<p>Nombre: Warning</p>	
<p>Familia: Feedback</p>	
<p>Alias: Warning, Think Twice</p>	
<p>PROBLEMA</p>	
<p>Qué información debe ser obtenida y especificada para solicitar al usuario la confirmación de la acción solicitada en caso de que ésta tenga consecuencias irreversibles.</p>	
<p>CONTEXTO</p>	
<p>Cuando una acción que tiene serias consecuencias es solicitada por el usuario.</p>	
<p>SOLUCIÓN</p>	
<p>Guía de Obtención de los Mecanismos de Usabilidad:</p>	
<p>Recomendación IPO</p>	<p>Temas a discutir con los stakeholders</p>
<p>1. Para cada acción que el usuario pueda ejecutar hay que considerar lo siguiente: si la acción es reversible, la proporción</p>	<p>1.1 Discutir con el usuario todas las acciones que se pueden realizar y sus consecuencias (considerar la</p>

<p>de acciones reversibles que el sistema soporta, la frecuencia con la que la acción se ejecuta, el grado de impacto que puede causar, y la inmediatez de la retroalimentación. Con todo esto se debe considerar que sería más apropiado, si un aviso, una confirmación o una autorización.</p>	<p>frecuencia de estas acciones y su impacto) y cuáles requieren confirmación (tener en cuenta no sobrecargar al usuario con avisos).</p>
<p>2. Puede que los usuarios no entiendan las consecuencias de sus acciones. Por eso, el aviso debería contener los siguientes elementos:</p> <ul style="list-style-type: none"> • Un resumen del problema y la condición que ha disparado el aviso. • Una pregunta para que el usuario indique si desea continuar con la acción o realizar otras acciones. Debe haber dos posibles elecciones para el usuario, seguir con la acción o abandonarla. • También puede incluir una descripción más detallada de la situación para ayudar al usuario a tomar la decisión. Las opciones deberían incluir un verbo que se refiera a la acción solicitada. • En algunos casos pueden haber más de dos opciones. El aumento del número de opciones se puede aceptar en algunos casos, pero se debe tender a minimizar. 	<p>2.1. ¿Qué información se mostrará para cada una de las acciones a confirmar? Se debe recordar que se deben proporcionar las consecuencias de la acción y las alternativas que se le darán al usuario.</p>
<p>Guía de Especificación de Mecanismos de Usabilidad:</p>	
<p>La siguiente información deberá ser instanciada en el documento de requisitos: - Las tareas U, V, Z necesitarán aviso. Para ellas, la información a mostrar será R, S, T respectivamente.</p>	

2. Wizard

2.1 Step by Step

IDENTIFICADOR	
Nombre: Step by Step	
Familia: Wizard	
Alias: Wizard Step by Step	
PROBLEMA	
Qué información se debe elicitarse y especificar para proporcionar a los usuarios el mecanismo de paso a paso.	
CONTEXTO	
Cuando un usuario novel necesita realizar una acción compleja y poco frecuente que consiste en varios pasos en los cuales se deben tomar ciertas decisiones.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Guiar al usuario a través de toda la tarea [Welie, 2000][Tidwell, 2005]. Cuando la tarea se inicia, el usuario debe ser informado sobre el objetivo que se pretende conseguir y las decisiones a tomar. Si es necesaria información del usuario, se le debe pedir de manera breve y sencilla en cada uno de los pasos en los que se divide la tarea.	1.1 ¿Qué tareas requieren varios pasos para ejecutarse? ¿Cuáles son esos pasos? 1.2 ¿Qué información es necesaria para el usuario en cada tarea? ¿Cuál es la mejor forma para preguntar por ella?
2. La tarea se puede bifurcar como un diagrama de flujo dependiendo de la información que introduzca el usuario. El usuario no tiene porqué conocer todos los caminos existentes para ejecutar la tarea [Welie, 2000]. Sin embargo, el	2.1 ¿Cuántos pasos son necesarios para la tarea? 2.2 ¿Cuál es la mejor forma de presentar estos pasos al usuario?

<p>usuario debe saber en qué paso se encuentra, especialmente si hay más de 7 pasos [Tidwell, 2005]. Si hay más de 10 pasos, se debe separar la tarea en otras tareas con una secuencia de pasos más manejable de manera que no sea tan tediosa para el usuario [Tidwell, 2005]. Esta división se puede llevar a cabo en base a la temática. La parte más difícil es balancear el número de los pasos de cada tarea.</p>	
<p>3. El usuario debe poder revisar una decisión navegando al paso anterior [Welie, 2000]. Incluso debe poder volver al primer paso si es necesario [Tidwell, 2005]</p>	<p>3.1 ¿Cuáles serán las opciones de navegación hacia atrás en cada uno de los pasos, ya sea al paso anterior o al comienzo de todas las tareas?</p>
<p>Guía de Especificación de Mecanismos de Usabilidad:</p>	
<p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <ul style="list-style-type: none"> - Las tareas A, B, C requieren varios pasos para ejecutarse. Los pasos para A son U, V, R, y la información que se debe mostrar en cada paso es R, S, T respectivamente. Estos pasos se representarán con formato J. El procedimiento sería similar para las tareas B y C. 	

3. User Input Error Prevention

3.1 Structured Text Entry

IDENTIFICADOR	
Nombre: Structured Text Entry	
Familia: User Input Error Prevention	
Alias: Form/ Field validation	
PROBLEMA	
Qué información se debe elicitarse y especificar para proporcionar texto de entrada de manera estructurada.	
CONTEXTO	
Cuando el sistema sólo puede aceptar información de entrada con un formato específico.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Facilitar la inserción de información que requiera un formato específico. Utilizar campos de inserción de diferentes tamaños (por ejemplo, para indicar el número de decimales que se esperan en un campo). Se debe evitar restringir la entrada de información demasiado o el usuario no será capaz de introducir valores válidos. Se debe hacer un testeo con usuarios para verificar si las restricciones son demasiado fuertes [Tidwell, 2005]. Además, se deben proporcionar ejemplos y valores por defecto para que el usuario tenga una idea del formato con el que proporcionar	1.1 ¿Dónde se necesitará información del usuario en un formato específico? 1.2 ¿Cómo guiar al usuario en la introducción de información que requiere un formato específico? (valores por defecto, uso de checklists o comboboxes)

la información [Griffiths, 2002]	
Guía de Especificación de Mecanismos de Usabilidad:	
La siguiente información debería ser instanciada en el documento de captura de requisitos:	
- El usuario introducirá la información M y N en formato P y Q respectivamente. El sistema guiará al usuario en el uso del formato correcto.	

4. User Profile

4.1 Preferences

IDENTIFICADOR	
Nombre: Preferences	
Familia: User Input Error Prevention	
Alias: Preferences User preferences	
PROBLEMA	
Qué información se debe elicitarse y especificar para proporcionar al usuario el mecanismo de preferencias.	
CONTEXTO	
Cuando: <ul style="list-style-type: none"> • El sistema es muy complejo y gran parte de su funcionalidad puede ser adaptada a las preferencias del usuario. • El sistema será usado por usuarios con diferentes habilidades, culturas y gustos. • No se conoce lo suficiente sobre las preferencias del usuario para asumir valores por defecto que resultarán agradables. 	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Proporcionar un repositorio donde los usuarios puedan almacenar sus preferencias en lenguaje, fuente, iconos, color y sonido. Este repositorio debe tener la capacidad de cargar las preferencias almacenadas cuando el usuario lo solicite. La información almacenada en el repositorio debe ser por usuario, aunque varios usuarios accedan al sistema [Tidwell, 2005]. Además, se debe permitir que las preferencias de un usuario se	1.1 ¿El sistema proporcionará al usuario la oportunidad de establecer preferencias particulares? (colores, fuente, formato, modos de selección de funcionalidad? En ese caso, ¿cuáles? 1.2 ¿Serán las preferencias diferentes para cada usuario? 1.3 ¿Cuál es la mejor forma para mostrar y permitir al usuario elegir sus preferencias?

<p>conviertan en los valores por defecto para otros usuarios [Welie, 2000].</p>	
<p>2. Proporcionar preferencias agrupadas en conjuntos. De esta manera, el usuario elegiría todas las preferencias incluidas en un conjunto. Esto facilita la elección del usuario que no desea pasar mucho tiempo buscando una combinación de preferencias adecuadas [Tidwell, 2005]. Se deben considerar los siguientes aspectos a la hora de proporcionar los conjuntos de preferencias:</p> <ul style="list-style-type: none"> • Los lenguajes nativos son más preferidos que el inglés. • Daltonismo. • Problemas de visión (hay usuarios que tienen un pequeño porcentaje de ceguera). • Problemas de oído. • Problemas de movilidad en las manos. 	<p>2.1 ¿Es conveniente para la aplicación disponer de preferencias agrupadas en conjuntos?</p> <p>2.2 ¿Qué preferencias se agrupan en cada conjunto?</p> <p>2.3 ¿Cuál es la mejor forma de presentar y permitir a los usuarios elegir las preferencias agrupadas?</p>
<p>Guía de Especificación de Mecanismos de Usabilidad:</p>	
<p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <p>- Las opciones A, B, C serán establecidas por el usuario. Estas opciones se presentarán con formato F.</p>	

4.2 Personal Object Space

<p>IDENTIFICADOR</p>
<p>Nombre: Personal Object Space</p>
<p>Familia: User Profile</p>
<p>Alias: Personal Object Space</p>
<p>PROBLEMA</p>

<p>Qué información se debe elicitar y especificar para proporcionar al usuario el mecanismo de personalizar la localización de los elementos de la interfaz.</p>	
<p>CONTEXTO</p> <p>Cuando la interfaz del sistema es compleja y tiene muchos iconos que se pueden organizar de diferentes maneras.</p>	
<p>SOLUCIÓN</p>	
<p>Guía de Obtención de los Mecanismos de Usabilidad:</p>	
<p>Recomendación IPO</p>	<p>Temas a discutir con los stakeholders</p>
<p>1. El usuario debe poder organizar los elementos de la interfaz de manera que se ajusten a sus necesidades, ya que el usuario es el que mejor conoce su forma de trabajar. El usuario puede recordar rápidamente donde está cada elemento en la interfaz si es él mismo el que lo ha ubicado en ese sitio [Tidwell, 2005]. Es tedioso para el usuario ubicar todos los elementos por lo que se aconseja utilizar valores por defecto. Las acciones que se deben habilitar son: mover, agrupar, alinear, ordenar.</p>	<p>1.1 ¿Los usuarios podrán organizar la ubicación de los elementos visibles de las interfaces?</p> <p>1.2 ¿El sistema habilitará esta posibilidad de organización para todos los usuarios?</p> <p>1.3 ¿Qué elementos de la interfaz podrá ordenar el usuario?</p>
<p>Guía de Especificación de Mecanismos de Usabilidad:</p> <p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <p>- Las opciones A, B, C serán organizadas por el usuario. Estas opciones se presentarán con formato F.</p>	

4.3 Favourites

<p>IDENTIFICADOR</p>
<p>Nombre: Favourites</p>
<p>Familia: User Profile</p>
<p>Alias: Bookmarks Favourites</p>

PROBLEMA	
<p>Qué información se debe elicitar y especificar para proporcionar al usuario el mecanismo de favoritos.</p>	
CONTEXTO	
<p>En un sistema navegable, complejo y con gran cantidad de interfaces, el usuario debería ser capaz de moverse libremente a través de las interfaces mediante accesos que no existen por defecto en el sistema.</p>	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
<p>1. Permitir al usuario almacenar sus puntos de interés de manera que pueda volver a ellos en cualquier momento. El usuario debe ser capaz de etiquetar estos puntos para recordarlos con un nombre que le sea familiar. La lista de favoritos se debe almacenar y estar disponible cada vez que la solicite el usuario [Tidwell, 2005].</p>	<p>1.1 ¿El sistema permitirá al usuario almacenar los puntos de interés que considere interesantes? 1.2 En ese caso, ¿cuántos puntos de interés podrá almacenar?</p>
<p>2. Si la lista es larga se debe permitir al usuario estructurarla [Welie, 2000]. Al menos se debería proporcionar la posibilidad de ordenar la lista para clasificar las entradas en base a cierto criterio especificado por el usuario. En caso de que sea posible, también se debe proporcionar un mecanismo de agrupación de los elementos de la lista de favoritos [Tidwell, 2005].</p>	<p>2.1 ¿Cómo se va a almacenar la lista de puntos de interés?</p>
Guía de Especificación de Mecanismos de Usabilidad:	
<p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <ul style="list-style-type: none"> - El sistema permitirá a cada usuario almacenar X puntos de interés de los que haya visitado. Estas interfaces se presentarán con un formato F. 	

5. Undo y Cancel

5.1 Global Undo

IDENTIFICADOR	
Nombre: Global Undo	
Familia: Undo / Cancel	
Alias: Multi-level Undo Undo Global Undo Allow Undo	
PROBLEMA	
Qué información se debe elicitar y especificar para proporcionar al usuario el mecanismo de deshacer.	
CONTEXTO	
Cuando se construye un sistema interactivo con funcionalidades múltiples y complejas.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Los usuarios exploran la funcionalidad de un sistema pero no quieren ser castigados si ejecutan una acción no deseada [Welie, 2000]. La capacidad de deshacer una secuencia de acciones hace más segura la exploración de las interfaces. Mientras el usuario aprende a interactuar con las interfaces, puede experimentar con ellas confiando en que podrá deshacer todas las acciones que realice. Es importante decidir qué acciones deben tener la capacidad de deshacerse [Tidwell, 2005]. Son susceptibles de recibir esta capacidad:	1.1 ¿Qué acciones serán permanentes? 1.2 De estas acciones permanente, ¿Cuáles tendrá sentido que tengan la capacidad de deshacerse?

<ul style="list-style-type: none">• Acciones que puedan cambiar un fichero (o repositorio) deben tener la capacidad de deshacerse. Más concretamente, estas acciones son:<ul style="list-style-type: none">○ La entrada de texto○ Transacciones en la base de datos○ Modificación de imágenes o dibujos○ Modificación de las opciones de visualización (posición, tamaño, agrupación, etc.)○ Operaciones con los ficheros○ Creación y modificación de elementos○ Operaciones de cortar y pegar• Hay otras acciones que generalmente no tiene sentido deshacer porque llenan la pila de elementos que no requieren ser deshechos. Estas acciones son:<ul style="list-style-type: none">○ Selección de texto u objetos○ Navegación entre interfaces○ Localización del ratón y el cursor○ Posición de la barra de desplazamiento○ Posición y tamaño de las interfaces○ Cambios hechos en una ventana modal antes de que se hayan confirmado dichos cambios• Si una acción tiene efectos secundarios que no se pueden deshacer se debe avisar al usuario sobre las posibles consecuencias	
--	--

<p>antes de ejecutar la acción.</p> <ul style="list-style-type: none"> • Se debe habilitar la capacidad de deshacer para aquellas acciones que tenga sentido para el usuario. Estas acciones pueden ser atómicas o un conjunto que agrupa varias [Welie, 2000]. Por ejemplo, la acción de deshacer la inserción de texto se debe hacer palabra por palabra y no letra por letra. 	
<p>2. Los usuarios suelen explorar todas las posibles navegaciones entre interfaces [Tidwell, 2005]. Es necesario una pila para almacenar las navegaciones que haga el usuario y deshacerlas cuando el usuario lo desee. La capacidad de deshacer se debe complementar con la capacidad de rehacer para que el usuario pueda volver a hacer las acciones que haya deshecho previamente. Las acciones de rehacer se almacenan también en una pila.</p>	<p>2.1 ¿El sistema requerirá diferentes combinaciones de la capacidad de deshacer/rehacer o sólo la de deshacer?</p> <p>2.2 ¿Con qué tipo de estructura se va a implementar la pila de acciones a deshacer/rehacer?</p> <p>2.3 ¿Cuántas acciones se almacenarán en la pila de deshacer/rehacer? Se debe decidir si el tamaño de la pila será siempre el mismo o variará en base al tipo de acciones que almacene.</p>
<p>3. La mayoría de las aplicaciones de Escritorio acceden a la funcionalidad de deshacer a través del menú Edición y mediante las teclas de acceso rápido CTRL+Z. Se debe mostrar el histórico de acciones de manera que el usuario sepa lo que ha hecho recientemente [Welie, 2000]. Este histórico muestra además cuál es la siguiente acción que se podría deshacer.</p>	<p>3.1 ¿Cuál es la mejor forma de presentar la funcionalidad de deshacer/rehacer al usuario?</p>
<p>Guía de Especificación de Mecanismos de Usabilidad:</p>	
<p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <p>- Las acciones U, V, Z se podrán deshacer. La información a almacenar en la pila es A,</p>	

B, C.

- La funcionalidad de deshacer y rehacer almacenará X acciones y la información almacenada se presentará con formato T.

5.2 Abort Operation

IDENTIFICADOR	
Nombre: Abort Operation	
Familia: Undo / Cancel	
Alias: Emergency exit Go back to a safe place Go back Prominent cancel	
PROBLEMA	
Qué información se debe elicitarse y especificar para proporcionar al usuario el mecanismo de abortar operación.	
CONTEXTO	
Cuando el usuario debe salir de una aplicación o abortar la ejecución de un servicio.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Los usuarios pueden necesitar salir de un programa cuando otro de mayor prioridad necesita recursos o cuando el programa se ha lanzado por error [Griffiths, 2002]. Por lo tanto, la opción de salir del programa debe estar disponible en todo momento, incluso si la interacción se produce en una ventana bloqueante. Si el usuario solicita la opción de salir y quedan cambios por guardar, la aplicación debería preguntar al usuario si desea guardar dichos cambios.	1.1 ¿Necesitará el usuario la opción de salir de la aplicación? 1.2 En caso afirmativo, ¿Dónde y cómo se mostrará esta opción al usuario?
2. En caso de que el usuario se encuentre	2.1 ¿Qué servicios requerirán una opción

<p>en una ventana en la cual no desea estar, se le debe proporcionar la posibilidad de volver a una ventana que le sea familiar. Las acciones de volver hacia atrás se pueden hacer pesadas en caso de que requieran navegar a través de muchas ventanas [Tidwell, 2005]. Por lo tanto, se recomienda proporcionar un botón Cancelar para cerrar todas las ventanas abiertas y descartar los posibles cambios que el usuario haya hecho desde la última vez que se guardaron los cambios.</p>	<p>de Cancelar? Se debe prestar especial atención a aquellos servicios que requieran varios pasos. Además, los servicios en los que el usuario es preguntado o se solicita información deben ser potencialmente cancelables.</p> <p>2.2 ¿Cómo se presentará la opción de cancelar al usuario (se debe tener en cuenta que la forma más común es mediante un botón de cancelar)?</p> <p>2.3 ¿Cuál será el estado al que vaya el sistema cuando se ejecute la opción de cancelar?</p>
<p>3. En caso de que la ejecución de un servicio requiera más de 10 segundos se debe proporcionar la opción de cancelar la ejecución y volver a un estado anterior. [Nielsen, 1993]. Este mecanismo se debe presentar junto con <i>Progress Feedback</i>.</p>	<p>3.1 ¿Qué servicios necesitarán más de 10 segundos para su ejecución?</p> <p>3.2 ¿Cómo se va a relacionar este mecanismo con <i>Progress Feedback</i>?</p>
<p>Guía de Especificación de Mecanismos de Usabilidad:</p>	
<p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <ul style="list-style-type: none"> - Los servicios A, B, C, necesitarán varios pasos y se debe proporcionar un mecanismo para abortar su ejecución. - El servicio W requiere más de 10 segundos para acabar y por tanto se le debe proporcionar la opción de cancelarlo. 	

6. Help

6.1 Multilevel help

IDENTIFICADOR	
Nombre: Multilevel help	
Familia: Help	
Alias: Multilevel help	
PROBLEMA	
Qué información se debe elicitar y especificar para proporcionar al usuario el mecanismo de información multinivel.	
CONTEXTO	
Cuando la aplicación a desarrollar es compleja y pocos usuarios van a necesitar ayuda para interactuar con ella. El otro gran porcentaje de usuarios no necesitará ayuda y el proporcionársela puede ralentizar sus tareas. Por lo tanto, se deben satisfacer las necesidades de los usuarios noveles y de los expertos.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
<p>1. Crear mensajes de ayuda para distintos tipos de usuarios. Estos mensajes se pueden mostrar de distintas maneras:</p> <ul style="list-style-type: none"> Mostrando un mensaje junto a cada elemento que requiera ayuda. La longitud del mensaje debe ser corta o pocos usuarios la leerán. Utilizando <i>Tooltips</i> cuando elementos de la ventana requieren una breve descripción para explicar su funcionamiento. Se suelen utilizar para explicar el significado de iconos. Como inconvenientes del uso de <i>Tooltips</i> cabe destacar que sólo se muestran al pasar el 	<p>1.3 ¿Cómo de complejas serán las tareas a realizar en el sistema?</p> <p>1.4 ¿El sistema lo usarán normalmente usuarios noveles o expertos?</p> <p>1.5 ¿Qué tareas necesitarán ayuda?</p> <p>1.6 ¿Qué tipo de ayuda es la más adecuada para cada tarea?</p>

<p>puntero del ratón sobre ellos y que algunos usuarios los pueden encontrar irritantes.</p> <ul style="list-style-type: none">• Descripción larga que se muestra dinámicamente cuando el usuario selecciona o pasa el puntero del ratón sobre un elemento de la ventana. Se debe reservar un área de la ventana para mostrar mensajes de ayuda de este tipo.• Paneles con la capacidad de cerrarse que contienen mensajes de ayuda largos.• Mostrar la ayuda en una ventana nueva a través de un navegador Web o mediante WinHelp. Estos mensajes de ayuda son normalmente manuales en línea o libros electrónicos.• Soporte técnico vía correo electrónico, Web o telefónico.	
Guía de Especificación de Mecanismos de Usabilidad:	
<p>La siguiente información debería ser instanciada en el documento de captura de requisitos:</p> <ul style="list-style-type: none">- El sistema proporcionará ayuda al usuario- Para las tareas A, B, C se proporcionará ayuda mediante <i>Tooltips</i> con la información L- Para las tareas U, V la ayuda estará compuesta por la información I que se mostrará con el formato F.	

7. Commands aggregation

7.1 Commands aggregation

IDENTIFICADOR	
Nombre: Commands aggregation	
Familia: Commands aggregation	
Alias: Componed commands Macros	
PROBLEMA	
Qué información se debe elicitarse y especificar para proporcionar al usuario el mecanismo de agregación de comandos.	
CONTEXTO	
Cuando los servicios del sistema se pueden lanzar en base al uso de comandos que se componen de expresiones más simples que deben ser precisas y a la vez entendibles por el usuario.	
SOLUCIÓN	
Guía de Obtención de los Mecanismos de Usabilidad:	
Recomendación IPO	Temas a discutir con los stakeholders
1. Los usuarios expertos tienden a utilizar comandos lingüísticos en vez de ventanas visuales. Además, en algunas ocasiones, los servicios disponibles no conviene mostrarlos gráficamente (por ejemplo, cuando hay un gran número de ellos). Para estos casos se debería proporcionar un mecanismo al usuario para que introduzca el comando directamente a través de la voz o mediante escritura. La retroalimentación al usuario sobre el resultado de la ejecución del servicio a través de los comandos debe ser inmediata. Además, las reglas	1.1 ¿Qué servicios se podrán agregar? 1.2 ¿Cómo se introducirán los comandos, por voz o por teclado? 1.3 ¿Cómo será la sintaxis para las agregaciones? Se debe usar una sintaxis clara y fácil de aprender.

sintácticas para construir los comandos deben ser sencillas de aprender y con un significado obvio.	
Guía de Especificación de Mecanismos de Usabilidad:	
La siguiente información debería ser instanciada en el documento de captura de requisitos: - Los servicios U, V, Z se podrán expresar mediante la agregación de comandos. Para los servicios U y V la agregación de comandos será mediante texto, mientras que para Z será por voz. La sintaxis utilizada para la agregación será S.	

Anexo II: Datos Recogidos en el Experimento

Toda la información mostrada en este Anexo se ha obtenido a partir de la página desde la que se hizo el experimento (<http://hci.dsic.upv.es/TareasEvaluacion>). En la Tabla Anexoll.1 se muestra la lista de participantes del experimento con su información personal: un identificador interno, el sexo, la ocupación actual, si tenía experiencia en el uso de aplicaciones Web, si tenía experiencia en el uso de aplicaciones desarrolladas con OlivaNOVA. Además también se muestra el tipo de aplicación con la que interactuó en el experimento. SM significa *Sin Mecanismos* y CM *Con Mecanismos*.

ID	Sexo	Edad	Ocupación	AplicacionWeb	OlivaNOVA	Aplicación
1	M	30	Investigadora en formación	si	si	SM
2	H	32	Investigador (doctorando)	si	si	CM
3	H	27	Investigador	si	si	SM
4	H	34	Profesor Universitario	si	si	CM
5	H	47	Profesor	si	si	SM
6	H	24	Estudiante	si	si	CM
7	M	37	Profesor	si	si	SM
8	M	23	investigador	si	no	SM
9	H	27	Becario	si	no	SM
10	M	30	Ingeniera Informática	si	si	SM
11	M	56	profesora	si	si	CM
12	H	27	I+D	si	no	CM
13	H	28	estudiante	si	no	CM
14	H	27	Ingeniero de Caminos	si	no	SM
15	H	32	Profesor	si	si	SM
16	H	27	Investigador	si	no	CM

ID	Sexo	Edad	Ocupación	AplicacionWeb	OlivaNOVA	Aplicación
17	M	25	Auxiliar de turismo	si	no	CM
18	H	27	Analyst Programmer	si	no	SM
19	H	32	técnico	si	si	SM
20	H	31	profesor	si	no	CM
21	H	30	Ingeniero Informático	si	si	CM
22	H	26	Investigador	si	si	SM
23	M	31	Informática	si	si	CM
24	H	21	Estudiante	no	no	SM
25	H	22	Estudiante	si	no	CM
26	H	25	Investigador/ Estudiante /Desarrollador	si	no	CM
27	H	27	Investigador	si	no	SM
28	H	55	Pensionista	si	no	SM
29	H	27	Programador	si	no	CM
30	H	22	Estudiante universitario	no	no	CM
31	H	46	Profesor	si	no	CM
32	M	21	Estudiante	no	no	SM
33	M	24	Investigación	si	no	SM
34	M	32	Profesora UPV	si	no	SM
35	H	31	Investigador	si	no	CM
36	H	27	Programador	si	si	CM
37	H	26	Estudiante	si	no	SM
38	M	28	investigador	no	si	CM
39	H	30	Ingeniero Industrial	si	no	CM
40	H	28	Profesor ESO y bachillerato	si	no	SM
41	M	44	semifuncionaria	si	no	CM
42	H	27	Ingeniero informático	si	si	SM

ID	Sexo	Edad	Ocupación	AplicacionWeb	OlivaNOVA	Aplicación
43	H	28	medico	si	no	SM
44	H	53	Ingeniero	si	no	SM
45	M	37	Profesora de Universidad	si	no	CM
46	H	30	Ingeniero Informatico	si	si	SM
47	M	30	Ingeniera Informática	si	si	CM
48	H	44	Electricista	si	no	CM
49	H	31	estudiante	si	no	SM
50	H	21	Estudiante	si	no	CM
51	H	20	Estudiante	si	no	SM
52	M	28	programación	si	no	SM
53	H	25	Estudiante	si	no	SM
54	H	29	Investigador	si	no	CM
55	H	36	Profesor	si	no	CM
56	M	36	Informática	si	no	SM
57	M	25	Investigadora ProS	si	si	CM
58	M	26	Becaria de investigación	si	si	SM
59	M	23	estudiante	si	no	CM
60	H	49	Ingeniero	si	no	SM
61	M	26	Contable	si	no	CM
62	H	33	Ingeniero Informático	si	no	SM
63	M	30	investigador	si	no	CM
64	M	40	Profesora de Universidad	si	no	SM
65	M	38	Profesor Universitario	si	no	CM
66	H	25	Doctorando	si	si	CM

Tabla Anexo II.1 Datos demográficos de los sujetos del experimento

La Tabla Anexo II.2 muestra el tiempo que dedicó cada usuario a desempeñar las tareas. Los campos InicioTX muestran la hora en la que inicio la tarea X y los campos FinTX muestran la hora en la que finalizó la tarea X. El tiempo entre tareas no se almacenó porque era el tiempo dedicado a rellenar el cuestionario y esa información es irrelevante.

ID	InicioT1	FinT1	InicioT2	FinT2	InicioT3	FinT3	InicioT4	FinT4
1	10:39:15	10:46:05	10:51:50	10:56:08	10:57:28	11:00:42	11:03:31	11:05:36
2	10:46:39	10:51:47	10:55:46	10:59:47	11:00:54	11:05:14	11:05:49	11:08:16
3	12:21:56	12:24:35	12:25:44	12:27:27	12:28:30	12:30:23	12:31:03	12:32:22
4	19:24:58	19:27:33	19:28:54	19:31:36	19:33:25	19:35:43	19:36:02	19:37:55
5	2:17:09	2:20:07	2:21:03	2:21:25	2:22:13	2:28:01	2:28:19	2:29:53
6	9:03:18	9:08:13	9:08:48	9:10:53	9:11:26	9:13:25	9:13:41	9:15:04
7	10:13:56	10:17:52	10:18:59	10:21:39	10:22:58	10:26:34	10:26:50	10:31:14
8	11:20:35	11:22:54	11:23:43	11:26:18	11:27:10	11:29:41	11:30:23	11:31:50
9	12:34:39	12:37:28	12:39:15	12:42:06	12:42:33	12:46:46	12:47:12	12:48:30
10	13:17:36	13:20:47	13:22:59	13:26:14	13:27:29	13:30:04	13:30:45	13:32:52
11	13:39:18	13:42:08	13:43:40	13:48:50	13:49:10	13:51:23	13:51:37	13:52:55
12	15:45:08	15:47:39	15:49:52	15:53:17	15:53:53	15:56:25	15:56:41	15:58:25
13	15:12:10	15:14:07	15:14:51	15:26:15	15:26:58	15:32:11	15:32:29	15:37:33
14	18:00:23	18:11:26	18:13:49	18:19:41	18:20:46	18:34:55	18:35:29	18:37:18
15	21:10:00	21:12:42	21:13:28	21:16:02	21:16:46	21:19:43	21:19:52	21:21:57
16	9:29:41	9:32:36	9:34:41	9:39:54	9:40:27	9:42:20	9:42:33	9:44:15
17	19:29:44	19:32:13	19:33:38	19:36:14	19:36:41	19:40:18	19:40:39	19:42:13
18	22:34:59	22:38:09	22:40:05	22:43:50	22:44:52	22:46:29	22:47:48	22:53:33
19	12:59:05	13:00:56	13:01:45	13:03:21	13:03:54	13:05:30	13:07:25	13:09:26
20	21:41:47	21:44:21	21:45:23	21:48:30	21:49:04	21:53:13	21:53:36	21:55:14
21	9:43:52	9:45:49	9:48:18	9:48:47	9:54:00	9:55:34	9:55:51	9:58:02
22	9:49:17	9:52:05	9:53:44	9:55:49	9:56:15	9:58:48	9:59:04	10:01:07
23	14:41:31	14:43:57	14:45:57	14:51:31	14:53:22	14:58:10	14:59:08	15:02:28
24	14:04:13	14:08:29	14:10:14	14:16:05	14:16:56	14:19:11	14:19:38	14:22:02
25	0:37:34	0:43:59	0:46:50	0:54:09	0:55:14	0:57:42	0:57:57	1:01:45
26	14:24:59	14:26:31	14:28:04	14:30:09	14:30:45	14:33:53	14:34:20	14:37:20
27	16:17:03	16:21:23	16:24:28	16:28:09	16:28:58	16:31:51	16:32:38	16:35:49
28	18:09:52	18:12:25	18:15:17	18:15:55	18:17:31	18:18:00	18:18:34	18:21:13
29	19:32:29	19:37:12	19:38:30	19:41:25	19:41:54	19:42:02	19:44:54	19:46:14

ID	InicioT1	FinT1	InicioT2	FinT2	InicioT3	FinT3	InicioT4	FinT4
30	22:38:28	22:42:16	22:43:38	22:52:57	22:53:25	22:58:57	22:59:33	23:03:19
31	18:57:57	19:00:12	19:01:42	19:05:50	19:06:17	19:08:36	19:08:49	19:10:59
32	14:57:02	15:00:52	15:02:12	15:06:17	15:06:49	15:11:07	15:11:22	15:14:31
33	10:33:11	10:36:27	10:38:18	10:41:56	10:43:03	10:46:36	10:46:48	10:48:09
34	13:28:22	13:31:38	13:33:34	13:35:46	13:51:04	13:52:19	13:52:47	13:53:52
35	13:32:58	13:35:07	13:36:56	13:39:41	13:40:13	13:42:08	13:42:31	13:44:19
36	16:09:53	16:11:07	16:12:01	16:14:56	16:15:29	16:19:59	16:20:14	16:25:45
37	15:51:13	15:53:33	15:56:26	16:00:01	16:01:13	16:03:34	16:04:13	16:06:11
38	16:20:17	16:25:16	16:30:47	16:36:27	16:38:39	16:44:54	16:46:41	16:49:56
39	17:51:27	17:53:26	17:54:38	17:58:23	17:58:56	18:00:24	18:00:34	18:04:03
40	17:52:54	17:56:59	17:58:43	18:04:15	18:04:41	18:09:20	18:09:29	18:11:01
41	22:56:04	22:56:54	22:59:56	23:01:03	23:02:26	23:03:43	23:04:35	23:06:15
42	17:13:54	17:14:39	17:20:39	17:25:47	17:26:33	17:30:34	17:30:54	17:32:10
43	18:20:05	18:22:10	18:23:58	18:24:23	18:25:06	18:25:30	18:25:54	18:26:55
44	3:30:22	3:35:25	3:36:48	3:45:13	3:45:54	3:53:42	3:54:29	4:00:41
45	7:02:23	7:05:11	7:06:23	7:11:40	7:12:13	7:18:33	7:18:48	7:21:31
46	10:32:50	10:34:43	10:35:53	10:38:38	10:39:03	10:41:11	10:41:58	10:43:35
47	12:30:36	12:33:17	12:34:53	12:38:27	12:39:30	12:41:19	12:41:49	12:44:30
48	11:39:20	11:44:40	11:48:32	11:57:38	11:58:30	12:09:12	12:10:04	12:20:52
49	9:38:38	9:45:29	9:49:53	9:53:49	9:54:27	10:00:30	10:00:43	10:09:07
50	16:55:36	16:58:20	17:00:03	17:06:27	17:06:59	17:09:25	17:09:56	17:11:37
51	13:03:04	13:05:14	13:07:28	13:09:38	13:10:56	13:13:03	13:13:24	13:15:38
52	21:50:59	21:55:01	21:59:32	22:03:08	22:04:53	22:09:50	22:10:31	22:16:31
53	13:22:25	13:26:19	13:28:07	13:30:09	13:30:53	13:33:07	13:33:26	13:35:04
54	12:23:16	12:25:19	12:27:17	12:28:46	12:31:35	12:33:32	12:34:08	12:36:04
55	17:33:05	17:34:14	17:35:51	17:42:39	17:43:46	17:46:04	17:46:17	17:47:57
56	21:54:19	21:59:10	22:02:40	22:08:30	22:09:32	22:12:13	22:13:14	22:16:32
57	10:32:17	10:36:15	10:38:20	10:40:44	10:41:12	10:45:25	10:45:50	10:49:13
58	13:03:35	13:06:53	13:08:19	13:12:29	13:13:14	13:16:33	13:16:52	13:18:20
59	15:25:04	15:28:30	15:29:44	15:37:02	15:37:59	15:48:11	15:48:29	15:51:06
60	18:02:21	18:05:22	18:06:47	18:12:17	18:14:39	18:22:01	18:22:16	18:23:15
61	23:08:43	23:12:04	23:13:06	23:18:31	23:18:50	23:26:35	23:27:01	23:29:10
62	23:36:58	23:40:59	23:43:03	23:43:45	23:44:47	23:50:34	23:51:29	23:54:55
63	10:49:39	10:52:06	10:54:08	10:59:34	11:00:56	11:04:59	11:05:34	11:07:55
64	18:02:18	18:07:34	18:10:23	18:19:45	18:20:52	18:24:09	18:24:25	18:26:09

ID	InicioT1	FinT1	InicioT2	FinT2	InicioT3	FinT3	InicioT4	FinT4
65	20:08:00	20:13:02	20:14:15	20:21:50	20:22:32	20:31:24	20:32:03	20:33:41
66	9:52:04	9:55:38	9:57:04	9:59:21	10:00:00	10:03:55	10:04:18	10:05:11

Tabla Anexo II.2 Datos con los tiempos por tarea

La Tabla Anexo II.3, Tabla Anexo II.4, Tabla Anexo II.5 muestran los resultados que cada usuario marcó en el cuestionario donde se recogía la satisfacción. Cada columna representa una pregunta del cuestionario. Los números de las tablas tienen el siguiente significado:

- 1: El usuario está totalmente de acuerdo con la sentencia en afirmativo.
- 2: El usuario está bastante de acuerdo con la sentencia en afirmativo.
- 3: El usuario no lo tiene claro o está indeciso, término medio.
- 4: El usuario está bastante de acuerdo con la sentencia en negativo.
- 5: El usuario está totalmente de acuerdo con la sentencia en negativo.

ID	P1	P2	P3	P4	P5	P6	P7
1	1	5	4	2	3	3	3
2	1	1	3	1	2	2	1
3	1	4	4	2	3	4	2
4	5	5	2	4	5	5	1
5	1	2	2	2	2	3	3
6	1	1	1	1	1	3	1
7	5	4	4	2	5	5	3
8	4	1	2	1	1	3	1
9	4	4	5	3	3	3	3
10	1	2	5	1	5	3	1
11	1	1	3	1	1	3	3
12	4	4	3	1	3	4	4
13	2	1	1	1	1	1	1

ID	P1	P2	P3	P4	P5	P6	P7
14	1	1	2	1	3	4	3
15	2	2	2	1	1	2	1
16	4	4	1	1	4	1	1
17	1	1	1	1	1	1	1
18	2	2	3	1	2	3	3
19	1	2	4	1	2	3	2
20	4	4	1	1	2	2	3
21	1	1	1	1	4	3	4
22	2	3	5	2	4	5	5
23	1	1	2	2	4	1	1
24	2	2	1	1	2	3	1
25	4	3	1	3	1	2	1
26	4	4	4	5	4	3	4
27	4	4	2	1	1	4	2
28	2	3	2	3	2	2	1
29	3	5	3	1	4	4	3
30	2	3	2	2	1	4	3
31	2	1	2	2	4	3	2
32	2	4	2	3	2	5	2
33	1	1	2	1	2	2	1
34	4	4	2	2	2	4	3
35	2	2	2	1	2	4	2
36	4	4	2	2	3	2	2
37	1	2	3	1	3	2	1
38	1	1	3	4	4	4	2
39	4	4	1	1	1	1	1
40	3	4	4	2	3	4	4
41	1	1	3	3	3	3	3
42	3	3	3	2	3	4	4
43	5	4	3	2	4	4	3
44	1	1	2	2	1	2	2
45	1	3	1	1	1	3	3
46	3	4	3	2	3	3	3
47	1	4	1	1	2	1	1
48	1	1	1	1	3	3	1

ID	P1	P2	P3	P4	P5	P6	P7
49	1	2	5	4	5	5	1
50	4	4	2	1	1	3	1
51	4	4	2	1	2	3	3
52	5	5	4	1	4	4	2
53	2	4	5	3	5	5	3
54	1	1	4	2	4	1	2
55	1	3	1	1	4	1	1
56	1	2	5	1	2	5	5
57	5	5	1	1	2	3	1
58	2	2	1	5	1	1	4
59	1	2	1	1	1	1	1
60	5	5	3	2	4	4	3
61	1	1	1	2	1	1	1
62	5	4	1	2	2	5	2
63	1	1	1	2	2	3	2
64	2	2	5	1	5	5	2
65	5	5	5	3	4	3	1
66	1	3	2	1	2	3	3

Tabla Anexo II.3 Respuestas de los usuarios al cuestionario de satisfacción (1)

ID	P8	P9	P10	P11	P12	P13	P14
1	3	3	3	3	3	2	1
2	1	1	2	1	2	1	1
3	5	3	3	3	3	3	3
4	5	1	5	5	5	1	3
5	3	3	3	3	3	4	4
6	1	1	4	2	3	1	1
7	5	5	5	5	5	5	3
8	2	5	4	4	4	3	2
9	5	5	3	3	3	5	3
10	4	5	5	3	3	5	3
11	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1
13	1	1	2	1	2	1	1

ID	P8	P9	P10	P11	P12	P13	P14
14	4	4	5	5	5	5	3
15	5	2	5	5	5	5	5
16	2	2	2	2	3	2	2
17	1	1	1	1	1	1	1
18	5	5	5	3	3	5	3
19	4	4	3	3	3	5	4
20	1	1	1	4	4	5	4
21	3	4	2	2	4	3	3
22	5	5	5	4	5	5	5
23	5	5	2	3	1	2	1
24	4	4	3	4	3	3	3
25	2	2	1	1	2	1	1
26	4	5	5	5	4	4	4
27	5	5	5	5	5	1	3
28	4	3	2	2	2	1	1
29	1	3	2	2	2	3	3
30	1	2	1	3	3	1	2
31	1	2	2	2	3	1	1
32	1	2	4	4	3	2	4
33	2	2	1	2	1	1	1
34	3	3	3	3	3	5	3
35	1	2	2	3	2	2	1
36	3	3	2	2	2	2	2
37	1	1	5	5	3	5	5
38	1	1	2	4	3	4	3
39	1	5	1	1	1	1	1
40	5	5	5	5	5	5	5
41	3	3	3	3	3	5	3
42	3	3	2	2	4	3	3
43	3	4	4	2	4	2	2
44	4	4	4	3	3	3	3
45	1	1	1	1	1	1	1
46	5	5	5	4	5	5	5
47	2	2	1	2	3	1	1
48	1	1	2	2	2	1	4

ID	P8	P9	P10	P11	P12	P13	P14
49	5	5	5	5	5	5	5
50	4	1	1	2	2	1	1
51	5	4	3	3	3	5	3
52	4	4	5	4	4	4	5
53	3	4	4	5	3	4	5
54	3	4	1	2	2	3	2
55	1	1	3	3	4	1	1
56	5	3	5	3	3	5	3
57	1	1	1	1	1	1	1
58	1	1	2	1	3	1	1
59	2	1	3	3	2	1	1
60	5	5	5	4	4	5	5
61	1	1	1	1	1	1	2
62	3	4	3	2	2	5	5
63	5	4	2	3	2	2	1
64	5	1	1	1	5	5	5
65	3	2	2	3	1	3	1
66	1	4	1	3	4	1	4

Tabla Anexo II.3 Respuestas de los usuarios al cuestionario de satisfacción (2)

ID	P15	P16	P17	P18	P19	P20	P21
1	1	3	1	4	2	3	2
2	1	2	1	1	2	2	1
3	3	5	2	5	2	4	4
4	2	1	5	1	5	5	5
5	4	4	2	4	3	3	3
6	1	1	5	1	2	4	3
7	3	1	1	5	4	5	5
8	4	2	2	2	3	2	3
9	3	1	1	1	2	3	2
10	3	1	1	1	4	3	4
11	1	2	2	2	1	1	1

ID	P15	P16	P17	P18	P19	P20	P21
12	1	2	5	1	2	3	2
13	1	1	1	1	2	3	2
14	3	1	1	1	4	4	5
15	5	3	2	3	2	2	2
16	2	2	5	3	5	5	4
17	1	1	2	1	1	1	1
18	2	1	1	4	2	3	3
19	3	1	1	2	3	3	3
20	4	3	1	4	4	4	4
21	3	1	4	2	2	2	2
22	5	1	2	5	3	4	4
23	1	2	3	3	2	2	2
24	3	1	1	1	2	2	1
25	1	1	1	1	1	2	1
26	3	1	1	5	4	4	4
27	3	1	2	3	4	4	3
28	2	2	2	1	1	1	1
29	3	2	1	1	4	4	4
30	2	3	4	4	1	2	2
31	1	1	2	1	2	2	2
32	3	1	1	1	2	2	2
33	2	1	1	2	1	1	1
34	1	1	1	5	2	3	2
35	1	1	4	1	2	2	2
36	2	2	4	4	3	5	4
37	1	1	1	5	3	3	2
38	2	5	5	5	3	4	3
39	1	3	3	3	2	2	2
40	5	2	3	4	4	4	4
41	3	5	5	5	5	5	5
42	2	1	1	2	2	3	3
43	3	3	2	2	2	2	3
44	3	1	2	1	2	3	2
45	1	1	1	1	1	1	1
46	5	1	1	1	3	3	3

ID	P15	P16	P17	P18	P19	P20	P21
47	1	2	5	2	2	3	2
48	2	1	2	2	2	2	2
49	5	5	5	4	5	5	5
50	3	1	1	1	2	3	2
51	3	1	1	1	3	3	2
52	1	2	1	1	3	4	4
53	3	1	1	5	2	3	2
54	3	2	5	2	3	4	3
55	1	3	2	5	5	5	4
56	3	1	1	1	1	1	2
57	1	5	5	5	2	3	3
58	1	3	5	2	1	1	1
59	1	3	2	2	1	1	1
60	5	5	5	5	4	4	4
61	1	1	1	1	1	1	1
62	4	1	1	1	4	3	3
63	1	5	5	5	3	3	4
64	5	1	1	5	3	5	5
65	1	1	1	2	5	4	4
66	2	1	3	1	2	2	2

Tabla Anexo II.3 Respuestas de los usuarios al cuestionario de satisfacción (3)

Índice de Figuras

Figura 2.1 Matriz que relaciona los beneficios en la usabilidad con los cambios que hay que añadir en la arquitectura del sistema en 27 escenarios [Bass, 2003].....	34
Figura 2.3 Tabla con los estilos de requisitos por cada característica de usabilidad [Lauesen, 1998].....	43
Figura 2.4 Parte del catálogo de usabilidad de Cysneiros [Cysneiros, 2003]..	49
Figura 2.5. Modelo de Usabilidad Propuesto por Adikari [Adikari, 2006].....	52
Figura 3.1 Relación entre atributos, propiedades y patrones de usabilidad....	103
Figura 3.2 Proceso de incorporación de mecanismos de usabilidad a los sistemas propuesto por Juristo	108
Figura 3.3 Vista esquemática de los componentes de un modelo conceptual	112
Figura 3.4 Transformaciones expresadas con un metamodelo	115
Figura 3.5 Modelos que componen el estándar MDA.....	116
Figura 3.6 Composición de patrones del Modelo de Interacción Abstracto [Molina, 2003]	129
Figura 3.7 Modelo de Interacción Concreto de OO-Method	131
Figura 3.8 Arquitectura Cliente / Servidor de OlivaNOVA.....	132
Figura 3.9 Clases generadas a partir de una UI de Servicio	133
Figura 3.10 Clases generadas a partir de una UI de Población	134
Figura 3.11 Clases generadas a partir de una UI de Instancia.....	134
Figura 3.12 Clases generadas a partir de una UI de Maestro/Detalle.....	135

Figura 3.13 Proceso de generación de código de OO-Method	136
Figura 4.1 Estructura de MIMAT	142
Figura 4.2 Resumen gráfico del método MIMAT	142
Figura 4.3 Representación de los mecanismos de usabilidad mediante Diagramas de Clase y de Secuencia	151
Figura 4.4 Tipos de representación de las clases al incorporar los mecanismos de usabilidad	152
Figura 4.5 Diagrama de Clases del caso de ilustración	156
Figura 5.1 Alta de cliente que informa sobre el éxito o fracaso de su ejecución.....	165
Figura 5.2 Mensaje de error para el servicio alquiler de coche	166
Figura 5.3 Mensaje de éxito para el servicio alquiler de coche	166
Figura 5.4 Reservar alquiler de coche donde se ve el estado de los coches.. ..	169
Figura 5.5 Impresión de facturas donde el botón de imprimir se inhabilita si no existen facturas	171
Figura 5.6 Mensaje de error por falta de recursos	174
Figura 5.7 Ejemplo de mensaje de error por un fallo en la comunicación con un dispositivo externo.....	176
Figura 5.8 Ejemplos de navegaciones con alias dinámicos.....	178
Figura 5.9 Diagrama de Clases para WU_SSF1 y WU_SSF3 de <i>System Status Feedback</i>	189
Figura 5.11 Diagrama de Clases para la WU_SSF2 de <i>System Status Feedback</i>	191
Figura 5.12 Diagrama de Secuencia para incorporar la forma de uso <i>Mostrar el estado de la información</i>	192
Figura 5.13 Diagrama de Secuencia para incorporar la forma de uso <i>Mostrar el estado de las acciones</i>	194

Figura 5.14 Reservar alquiler de coche con inhabilitación de componentes de la interfaz	202
Figura 5.15 Diagrama de Clases para <i>Interaction Feedback</i>	204
Figura 5.16 Diagrama de Secuencia para incorporar la forma de uso <i>Informar que la interacción está siendo atendida</i>	205
Figura 5.17 a) Barra de progreso sin mostrar tiempo restante textualmente. b) Barra de progreso mostrando el tiempo restante relativo a las tareas completadas por el sistema.....	211
Figura 5.18 Barra de progreso mostrando el tiempo real restante para finalizar la ejecución	211
Figura 5.19 Barra de progreso dentro de la ventana principal.....	211
Figura 5.20 a) Progreso textual mostrando la lista de tareas que componen el servicio. b) Progreso textual mostrando sólo la tarea que se está ejecutando y el tiempo relativo que resta para la finalización de la ejecución.	212
Figura 5.21 Diagrama de Clases para <i>Progress Feedback</i>	217
Figura 5.22 Diagrama de Secuencia para incorporar la forma de uso <i>Mostrar progreso de la ejecución</i> en una transacción global	219
Figura 5.23 Diagrama de Secuencia para incorporar la forma de uso <i>Mostrar progreso de la ejecución</i> en una transacción local	220
Figura 5.24 Ejemplo de ventana de Reservar alquiler de coche que muestra mensaje de aviso.....	226
Figura 5.25 Ejemplo de mensaje de aviso para el servicio Reservar alquiler de coche	226
Figura 5.26 Diagrama de Clases para <i>Warning</i>	231
Figura 5.27 Diagrama de Secuencia para incorporar la forma de uso <i>Mensaje de aviso</i>	231
Figura 6.1 Paso1: Identificar cliente	241
Figura 6.2 Paso 2: Introducir datos del nuevo cliente	241

Figura 6.3 Paso3: Introducir dirección del nuevo cliente	242
Figura 6.4 Paso 4: Introducir otros datos del cliente	242
Figura 6.5 Paso 5: Introducir datos bancarios.....	243
Figura 6.6 Paso 6: Introducir fechas del alquiler	243
Figura 6.7 Paso 7: Seleccionar coche.....	244
Figura 6.8 Diagrama de Clases para <i>Step by Step</i>	250
Figura 6.9 Diagrama de Secuencia para incorporar la forma de uso <i>Definir un asistente</i>	250
Figura 7.1 Ejemplo de tipos de valores para la propiedad <i>Tipo de campo de entrada</i>	257
Figura 7.2 Ejemplo de tipos de valores para la propiedad <i>Expresión regular</i>	258
Figura 7.2 Diagrama de Clases para <i>Structured Text Entry</i>	263
Figura 8.1 Listado de coches sin preferencias.....	272
Figura 8.2 Listado de coches con preferencias	273
Figura 8.3 Listado de coches cambiando el idioma	274
Figura 8.4 Diagrama de Clases para <i>Preferences</i>	282
Figura 8.5 Diagrama de Secuencia para incorporar la forma de uso <i>Preferencias en el aspecto visual</i>	283
Figura 8.6 Listado de coches aplicando las propiedades de <i>Distribución de elementos</i>	291
Figura 8.7 Diagrama de Clases para <i>Personal Object Space</i>	297
Figura 8.8 Diagrama de Secuencia para incorporar la forma de uso <i>Distribución de elementos</i>	297
Figura 8.9 Alta de cliente con la forma de uso <i>Definición de favoritos</i>	305

Figura 8.10 Alta de un nuevo favorito	305
Figura 8.11 Menú con la lista de favoritos.....	306
Figura 8.12 Diagrama de Clases para <i>Favourites</i>	309
Figura 8.13 Diagrama de Secuencia para incorporar la forma de uso <i>Definición de favoritos</i>	310
Figura 9.1 Poner coche en venta con la capacidad de deshacer	318
Figura 9.2 Listado de coches en venta	318
Figura 9.3 Diagrama de Clases para <i>Global Undo</i>	328
Figura 9.4 Diagrama de Secuencia para incorporar la forma de uso <i>Deshacer cambios</i> (almacenar antiguos valores para deshacer)	329
Figura 9.5 Diagrama de Secuencia para <i>Global Undo</i> (cargar antiguos valores al deshacer)	330
Figura 9.6 Diagrama de Secuencia para incorporar la forma de uso <i>Rehacer cambios</i> (cargar antiguos valores en la pila para rehacer)	332
Figura 9.7 Diagrama de Secuencia para incorporar la forma de uso <i>Rehacer cambios</i> (rehacer antiguos valores de la pila)	332
Figura 9.8 Ejecución de reservar alquiler de coche	340
Figura 9.9 Alta de cliente con capacidad para volver a reservar alquiler de coche	342
Figura 9.10 Diagrama de Clases para <i>Abort Operation</i>	348
Figura 9.11 Diagrama de Secuencia para incorporar la forma de uso <i>Cancelar durante la ejecución</i>	349
Figura 10.1 Poner en venta con ayuda dinámica.....	358
Figura 10.2 Diagrama de Clases para <i>Multilevel Help</i>	367
Figura 11.1 Modelo de Objetos para modelar SSF1 (1)	376
Figura 11.2 Modelo de Objetos para modelar SSF1 (2)	376

Figura 11.3 Modelo de Objetos para modelar WU_GU1 y WU_GU2.....	377
Figura 11.4 Modelo de Objetos para modelar WU_AO1	378
Figura 11.5 Modelo de Objetos para modelar WU_STE3.....	379
Figura 11.6 Modelo de Objetos para modelar WU_W1(1).....	380
Figura 11.7 Modelo de Objetos para modelar WU_W1(2).....	381
Figura 11.8 Modelo de Interacción Abstracto para modelar WU_SBS1 (1)	383
Figura 11.9 Modelo de Interacción Abstracto para modelar WU_SBS1 (2)	384
Figura 11.10 Modelo de Interacción Abstracto para modelar WU_PF1	385
Figura 11.11 Modelo de Interacción Abstracto para modelar WU_SSF3..	387
Figura 11.12 Modelo de Interacción Abstracto para modelar WU_STE2..	388
Figura 11.13 Modelo de Interacción Abstracto para modelar WU_MH1 ...	389
Figura 11.14 Modelado del Árbol de Jerarquía de Acciones	390
Figura 11.15 Añadir un nuevo elemento al Árbol de Jerarquía de Acciones...	391
Figura 11.16 Modelado del acceso a las funcionalidades de WU_GU1 y WU_GU2 a partir del menú del sistema (1)	391
Figura 11.17 Modelado del acceso a las funcionalidades de WU_GU1 y WU_GU2 a partir del menú del sistema (2)	392
Figura 11.18 Modelado del acceso a la funcionalidad WU_MH2	393
Figura 11.19 Modelado del acceso a la funcionalidad WU_F1.....	394
Figura 11.20 Modelo de Interacción Concreto para modelar WU_SBS1 ..	396
Figura 11.21 Modelo de Interacción Concreto para modelar WU_STE1...	397

Figura 11.22 Modelo de Interacción Concreto para modelar WU_P1 (texto)	398
Figura 11.23 Modelo de Interacción Concreto para modelar WU_P1 (iconos)	398
Figura 11.24 Modelo de Interacción Concreto para modelar WU_P1 (elementos).....	399
Figura 11.25 Modelo de Interacción Concreto para modelar WU_P1 (colores).....	400
Figura 11.26 Modelo de Interacción Concreto para modelar WU_POS1 ..	401
Figura 11.27 Modelo de Interacción Concreto para modelar WU_AO1 (1)	402
Figura 11.28 Modelo de Interacción Concreto para modelar WU_AO1 (2)	402
Figura 11.29 Modelo de Interacción Concreto para modelar WU_AO2 (1)	403
Figura 11.30 Modelo de Interacción Concreto para modelar WU_AO2 (2)	404
Figura 11.31 Modelo de Interacción Concreto para modelar WU_MH1	405
Figura 11.32 Modelo de Interacción Concreto para modelar WU_PF1 (1).....	406
Figura 11.33 Modelo de Interacción Concreto para modelar WU_PF1 (2).....	407
Figura 11.34 Modelo de Interacción Concreto para modelar WU_SSF1 (1) ...	408
Figura 11.35 Modelo de Interacción Concreto para modelar WU_SSF1 (2) ...	409
Figura 11.36 Modelo de Interacción Concreto para modelar WU_SSF2...	410
Figura 11.37 Modelo de Interacción Concreto para modelar WU_W1	411
Figura 11.38 Modelo de Interacción Concreto para modelar WU_F1	412

Figura 12.1 Proceso del experimento	428
Figura 12.2 Ventana para crear coche sin utilizar formas de uso.....	445
Figura 12.3 Ventana para crear coche utilizando formas de uso.....	446
Figura 12.4 Mensaje para confirmar que el servicio se ha ejecutado correctamente.....	447
Figura 12.5 Ventana para crear cuenta bancaria sin utilizar formas de uso....	448
Figura 12.6 Ventana para crear cuenta bancaria utilizando formas de uso	448
Figura 12.7 Ventana para hacer una reserva de alquiler sin utilizar formas de uso.....	449
Figura 12.8 Ventana para hacer una reserva de alquiler utilizando formas de uso	450
Figura 12.9 Mensaje de aviso para reservas de alquiler con más de un mes de duración.....	450
Figura 12.10 Ventana para poner un coche en venta sin utilizar formas de uso	451
Figura 12.11 Mensaje de error cuando el usuario solicita poner un coche en venta habiendo alquileres pendientes.....	451
Figura 12.12 Ventana para poner un coche en venta utilizando formas de uso	452
Figura 12.13 Comparativa de las medias aritméticas de las respuestas entre los cuatro tipos de usuarios.....	455
Figura 12.14 Comparativa de las medias aritméticas de las formas de uso entre los cuatro tipos de usuarios	461
Figura 12.15 Prueba K-S para WU_SSF1, WU_STE1, WU_STE3, WU_STE2.....	467
Figura 12.16 Prueba K-S para WU_MH1, WU_W1, WU_SSF3, Generales	467

Figura 12.17 Prueba de Levene para la satisfacción.....	469
Figura 12.18 ANOVA de la satisfacción con el factor experiencia.....	470
Figura 12.19 ANOVA de la satisfacción con el factor formas de uso	471
Figura 12.20 ANOVA de la satisfacción estudiando dos factores: experiencia y usabilidad.....	472
Figura 12.21 ANOVA de los distintos tipos de atributos con el factor experiencia	473
Figura 12.22 ANOVA de los distintos tipos de atributos con el factor formas de uso.....	474
Figura 12.23 ANOVA de los tres atributos estudiados con el factor experiencia	474
Figura 12.24 ANOVA de los tres atributos estudiados con el factor experiencia	475
Figura 12.25 Gráfico box and whiskers de SUMRespuestaWU_SSF1 con el factor experiencia	476
Figura 12.26 Gráfico box and whiskers de SUMRespuestaWU_SSF1 con el factor formas de uso.....	477
Figura 12.27 Gráfico box and whiskers de SUMRespuestaWU_W1 con el factor experiencia	478
Figura 12.28 Gráfico box and whiskers de SUMRespuestaWU_W1 con el factor formas de uso.....	479
Figura 12.29 Comparativa de las medias aritméticas de los tiempos por tarea entre los cuatro tipos de usuarios	481
Figura 12.30 Medias aritméticas de los tiempos totales de interacción de los cuatro tipos de usuarios	483
Figura 12.31 Prueba K-S para Tiempo_T1, Tiempo_T2, Tiempo_T3, Tiempo_T4, Tiempo_Total	485
Figura 12.32 Prueba de Levene para la eficiencia.....	486
Figura 12.33 ANOVA de la eficiencia con el factor experiencia.....	486

Figura 12.34 ANOVA de la eficiencia con el factor formas de uso 487

Figura 12.35 ANOVA de la eficiencia estudiando dos factores: experiencia y formas de uso 488

Figura 12.36 Gráfico box and whiskers de Tiempo_Total con el factor experiencia 488

Figura 12.37 Gráfico box and whiskers de Tiempo_Total con el factor formas de uso 489

Índice de Tablas

Tabla 1.1 Resultados del estudio de la complejidad de incorporar <i>Feedback</i> y <i>Cancel</i> a la arquitectura de un sistema	14
Tabla 2.1 Modelo de Usabilidad de Abrahao [Abrahao, 2006] [Abrahao, 2005] [España, 2006].	85
Tabla 2.2 Comparación entre todos los autores estudiados en el estado de la cuestión	96
Tabla 3.1 División de FUF en mecanismos de usabilidad [Juristo, 2007, b]	106
Tabla 5.1 Derivación de las formas de uso de <i>System Status Feedback</i> a partir de la guía de captura de requisitos	160
Tabla 5.2 Derivación de propiedades de la forma de uso <i>Informar del éxito o fracaso en la ejecución del servicio</i> a partir de la guía de captura de requisitos	163
Tabla 5.3 Derivación de propiedades de la forma de uso <i>Mostrar el estado de la información</i> a partir de la guía de captura de requisitos	167
Tabla 5.4 Derivación de propiedades de la forma de uso <i>Mostrar el estado de las acciones</i> a partir de patrones de usabilidad de Tidwell.....	170
Tabla 5.5 Derivación de propiedades de la forma de uso <i>Informar de falta de recursos</i> a partir de la guía de captura de requisitos	173
Tabla 5.6 Derivación de propiedades de la forma de uso <i>Informar de fallos en dispositivos externos</i> a partir de la guía de captura de requisitos	175
Tabla 5.7 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>System Status Feedback</i>	182
Tabla 5.8 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Informar del éxito o fracaso en la ejecución del servicio</i>	185
Tabla 5.9 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Mostrar el estado de la información</i>	186

Tabla 5.10 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Mostrar el estado de las acciones</i>	188
Figura 5.10 Diagrama de Secuencia para incorporar la forma de uso <i>Informar del éxito o fracaso en la ejecución del servicio</i>	190
Tabla 5.11 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Informar del éxito o fracaso en la ejecución del servicio</i>	191
Tabla 5.12 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Mostrar el estado de la información</i>	192
Tabla 5.13 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Mostrar el estado de las acciones</i>	195
Tabla 5.14 Resumen de los cambios para incorporar <i>System Status Feedback</i>	197
Tabla 5.15 Derivación de la forma de uso de <i>Interaction Feedback</i> a partir de la guía de captura de requisitos	198
Tabla 5.16 Derivación de propiedades de la forma de uso <i>Informar que la interacción está siendo atendida</i> a partir de la guía de captura de requisitos	200
Tabla 5.17 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Interaction Feedback</i>	203
Tabla 5.18 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Informar que la interacción está siendo atendida</i>	205
Tabla 5.19 Resumen de los cambios para incorporar <i>Interaction Feedback</i>	206
Tabla 5.20 Derivación de las formas de uso de <i>Progress Feedback</i> a partir de la guía de captura de requisitos	207
Tabla 5.21 Derivación de propiedades de la forma de uso <i>Mostrar progreso de la ejecución</i> a partir de la guía de captura de requisitos.....	208
Tabla 5.22 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Progress Feedback</i>	214

Tabla 5.23 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Mostrar progreso de la ejecución</i>	217
Tabla 5.24 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Mostrar progreso de la ejecución</i>	221
Tabla 5.25 Resumen de los cambios para incorporar <i>Progress Feedback</i>	222
Tabla 5.26 Derivación de la forma de uso de <i>Warning</i> a partir de la guía de captura de requisitos	223
Tabla 5.27 Derivación de propiedades de la forma de uso <i>Mensaje de aviso</i> a partir de la guía de captura de requisitos	224
Tabla 5.28 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Warning</i>	228
Tabla 5.29 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Mensaje de aviso</i>	230
Tabla 5.30 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Mensaje de aviso</i>	232
Tabla 5.31 Resumen de los cambios para incorporar <i>Warning</i>	234
Tabla 6.1 Derivación de la forma de uso de <i>Step by Step</i> a partir de la guía de captura de requisitos	236
Tabla 6.2 Derivación de propiedades de la forma de uso <i>Definir un asistente</i> a partir de la guía de captura de requisitos	238
Tabla 6.3 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Step by Step</i>	246
Tabla 6.4 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Definir un asistente</i>	248
Tabla 6.5 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Definir un asistente</i>	251
Tabla 6.6 Resumen de los cambios para incorporar <i>Step by Step</i>	252
Tabla 7.1 Derivación de las formas de uso de <i>Structured Text Entry</i> a partir de la guía de captura de requisitos	254

Tabla 7.2 Derivación de la propiedad de la forma de uso <i>Especificar el tipo de visualización del campo de entrada el tipo de entrada</i> a partir de la guía de captura de requisitos	256
Tabla 7.3 Derivación de propiedades de la forma de uso <i>Definición de máscaras</i> a partir de la guía de captura de requisitos	257
Tabla 7.4 Derivación de propiedades de la forma de uso <i>Valores por defecto</i> a partir de la guía de captura de requisitos	259
Tabla 7.5 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Structured Text Entry</i>	261
Tabla 7.6 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Tipo de campo de entrada</i>	262
Tabla 7.7 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Tipo de campo de entrada</i>	264
Tabla 7.8 Resumen de los cambios para incorporar <i>Structured Text Entry</i>	265
Tabla 8.1 Derivación de las formas de uso de <i>Preferences</i> a partir de la guía de captura de requisitos	268
Tabla 8.2 Derivación de propiedades de la forma de uso <i>Preferencias en el aspecto visual</i> a partir de la guía de captura de requisitos	270
Tabla 8.3 Derivación de la propiedad de la forma de uso <i>Preferencias en el idioma</i> a partir de la guía de captura de requisitos	273
Tabla 8.4 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Preferences</i>	277
Tabla 8.5 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Preferencias en el aspecto visual</i>	281
Tabla 8.6 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Preferencias en el aspecto visual</i>	284
Tabla 8.7 Resumen de los cambios para incorporar <i>Preferences</i>	286
Tabla 8.8 Derivación de la forma de uso de <i>Personal Object Space</i> a partir de la guía de captura de requisitos	287

Tabla 8.9 Derivación de propiedades de la forma de uso <i>Distribución de elementos</i> a partir de la guía de captura de requisitos	289
Tabla 8.10 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Personal Object Space</i>	293
Tabla 8.11 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Distribución de elementos</i>	296
Tabla 8.12 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Distribución de elementos</i>	299
Tabla 8.13 Resumen de los cambios para incorporar <i>Personal Object Space</i>	301
Tabla 8.14 Derivación de la forma de uso de <i>Favourites</i> a partir de la guía de captura de requisitos	302
Tabla 8.15 Derivación de las propiedad de la forma de uso <i>Definición de favoritos</i> a partir de la guía de captura de requisitos	303
Tabla 8.16 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Favourites</i>	307
Tabla 8.17 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Definición de favoritos</i>	308
Tabla 8.18 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Definición de Favoritos</i>	311
Tabla 8.19 Resumen de los cambios para incorporar <i>Favourites</i>	311
Tabla 9.1 Derivación de las formas de uso de <i>Global Undo</i> partir de la guía de captura de requisitos	314
Tabla 9.2 Derivación de propiedades de la forma de uso <i>Deshacer cambios</i> a partir de la guía de captura de requisitos.....	316
Tabla 9.3 Derivación de propiedades de la forma de uso <i>Rehacer cambios</i> a partir de la guía de captura de requisitos.....	319
Tabla 9.4 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Global Undo</i>	324

Tabla 9.5 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Deshacer cambios</i>	325
Tabla 9.6 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Rehacer cambios</i>	326
Tabla 9.7 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Deshacer cambios</i>	331
Tabla 9.8 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Rehacer cambios</i>	334
Tabla 9.9 Resumen de los cambios para incorporar <i>Global Undo</i>	336
Tabla 9.10 Derivación de las formas de uso de <i>Abort operation</i> a partir de la guía de captura de requisitos	337
Tabla 9.11 Derivación de propiedades de la forma de uso <i>Cancelar durante la ejecución</i> a partir de la guía de captura de requisitos.....	339
Tabla 9.12 Derivación de propiedades de la forma de uso <i>Salir de un escenario</i> a partir de la guía de captura de requisitos	341
Tabla 9.13 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Abort operation</i>	344
Tabla 9.14 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Cancelar durante la ejecución</i>	346
Tabla 9.15 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Salir de un escenario</i>	347
Tabla 9.16 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Cancelar durante la ejecución</i>	349
Tabla 9.17 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Salir de un escenario</i>	350
Tabla 9.18 Resumen de los cambios para incorporar <i>Abort Operation</i>	351
Tabla 10.1 Derivación de las formas de uso de <i>Multilevel help</i> a partir de la guía de captura de requisitos	354
Tabla 10.2 Derivación de propiedades de la forma de uso <i>Ayuda dinámica</i> a partir de la guía de captura de requisitos.....	356

Tabla 10.3 Derivación de propiedades de la forma de uso <i>Ayuda estática</i> a partir de la guía de captura de requisitos	359
Tabla 10.4 Resumen de los cambios en OO-Method que produce cada una de las propiedades de <i>Multilevel help</i>	363
Tabla 10.5 Resumen de las nuevas primitivas conceptuales para incorporar la forma de uso <i>Ayuda dinámica</i>	365
Tabla 10.6 Resumen de la nueva primitivas conceptual para incorporar la forma de uso <i>Ayuda estática</i>	366
Tabla 10.7 Resumen de los cambios en el compilador de modelos para incorporar la forma de uso <i>Ayuda dinámica</i>	368
Tabla 10.8 Resumen de los cambios para incorporar <i>Multilevel Help</i>	370
Tabla 12.1 Resumen de las formas de uso incorporadas en el experimento.. ..	421
Tabla 12.2 Objetivos específicos del experimento	422
Tabla 12.3 Diseño factorial 2x2 del experimento	427
Tabla 12.4 Relación entre <i>Informar del éxito o fracaso en la ejecución del servicio</i> , atributos y criterios ergonómicos	431
Tabla 12.5 Relación entre <i>Mostrar el estado de las acciones</i> , atributos y criterios ergonómicos	432
Tabla 12.6 Relación entre <i>Mensaje de aviso</i> , atributos y criterios ergonómicos	432
Tabla 12.7 Relación entre <i>Especificar el tipo de visualización del campo de entrada</i> , atributos y criterios ergonómicos	433
Tabla 12.8 Relación entre <i>Definición de máscaras</i> , atributos y criterios ergonómicos	434
Tabla 12.9 Relación entre <i>Valores por defecto</i> , atributos y criterios ergonómicos	434
Tabla 12.10 Relación entre <i>Ayuda dinámica</i> , atributos y criterios ergonómicos	434

Tabla 12.11 Ejemplo de pregunta del cuestionario.....	437
Tabla 12.12 Preguntas del cuestionario para la tarea 1	439
Tabla 12.13 Preguntas del cuestionario para la tarea 2	441
Tabla 12.14 Preguntas del cuestionario para la tarea 3	442
Tabla 12.15 Preguntas del cuestionario para la tarea 4	443
Tabla 12.16 Preguntas generales del cuestionario.....	444
Tabla 12.17 Tipos de usuario que mejor valoran las formas de uso	464
Tabla 12.18 Interpretación de la significancia.....	468
Tabla 12.19 Prueba de independencia de Durbin-Watson para la satisfacción.....	469
Tabla 12.20 Prueba de independencia de Durbin-Watson para la eficiencia.. ..	485
Tabla Anexo II.1 Datos demográficos de los sujetos del experimento	535
Tabla Anexo II.2 Datos con los tiempos por tarea	538
Tabla Anexo II.3 Respuestas de los usuarios al cuestionario de satisfacción (1)	540
Tabla Anexo II.3 Respuestas de los usuarios al cuestionario de satisfacción (2)	542
Tabla Anexo II.3 Respuestas de los usuarios al cuestionario de satisfacción (3)	544