UNIVERSIDAD DE VALENCIA
Facultad de Psicología

DESCRIPCION DE INTERFACES HOMBRE-ORDENADOR POR MEDIO DE METODOS FORMALES: APLICACION DE METODOS PARA LA EVALUACION DE UN INTERFAZ SIMULADO

TESIS DOCTORAL

Presentada por:

Pedro M. Valero Mora

Dirigida por:

Jaime Sanmartín Arce

Valencia, Febrero 1996

UNIVERSIDAD DE VALENCIA
Facultad de Psicología

DESCRIPCION DE INTERFACES HOMBRE-ORDENADOR POR MEDIO DE METODOS FORMALES: APLICACION DE METODOS PARA LA EVALUACION DE UN INTERFAZ SIMULADO

TESIS DOCTORAL

Presentada por:

Pedro M. Valero Mora

Dirigida por:

Jaime Sanmartín Arce

Valencia, Febrero 1996

A mi padre, a mi madre y a mi hermano.

Agradecimientos

Durante el proceso de elaboración de esta tesis doctoral, muchas personas aportaron apoyo, ideas o inspiración que, sin duda alguna, mejoraron este trabajo más allá de lo que el frugal talento de su autor habría sido capaz de producir sin su ayuda. A todos ellos, mi sincero reconocimiento, y mis disculpas a todos los que, aún sabiendo ellos que están en mi ánimo, no son nombrados a continuación.

En primer lugar, agradecerle al director de este trabajo, el doctor Jaime Sanmartín, la presencia y serenidad con las que ha acompañado el desarrollo de esta tesis, y que lo han convertido en un factor fundamental a la hora de explicar los aciertos que en ella puedan encontrarse. En segundo lugar, agradecer a Jesús Gabriel Molina, un compañero de trabajo que es sobre todo un amigo, el cual llevó a cabo con estoicismo una revisión que, sin duda, le convierte en merecedor de un especial reconocimiento.

En tercer lugar, me gustaría agradecer al doctor Andrew Monk, y a sus colaboradores, Leon Watts y Owen Daly-Jones, por las grandes facilidades que recibí en dos visitas a la Universidad de York. Gracias a ellos obtuve buena parte de la información y orientación utilizadas en este trabajo y, también, pero no menos importante, me sentí cómodo en un país distinto, alejado de mi ambiente habitual. Mi visión del mundo no sería la misma sin esta experiencia.

En cuarto lugar, agradecer a Tomás Martínez y a Fernando Canet por su ayuda en la ejecución de algunas de las partes de este trabajo.

En quinto lugar, agradecer al doctor Juan Luis Chorro Gascó ciertos comentarios acerca de cuestiones estadísticas, los cuales me llevaron al hallazgo de soluciones mucho más satisfactorias que las que había encontrado hasta ese momento. En unas breves conversaciones, casi sin darse cuenta del valor de sus palabras, hizo cambiar el enfoque de muchas de mis ideas y me obligó a esforzarme en la mejor dirección.

Un último grupo de personas a los que quiero mencionar en estos agradecimientos son todos los que, más comprometidos con el estado de mi ánimo que con la precisión de mi juicio, me auxiliaron en la difícil tarea de sentirme bien conmigo mismo. Son mis amigos y seres queridos. Entre todos me han transmitido ilusión, y han hecho que mis esfuerzos cotidianos se volvieran serenos y satisfactorios. Por todo ello, les doy las gracias.

Valencia, Enero de 1995

Indice

All correct reasoning is a grand system of tautologies, but only God could make direct use of that fact. The rest of us must painstakingly and fallibly tease out the consequences of our assumptions.

H. A. Simon. "The Sciences of the Artificial" p. 19

Todo razonamiento correcto es un gran sistema de tautologías, pero sólo Dios podría hacer uso de ese hecho directamente. El resto de nosotros debe desenredar las consecuencias de nuestras premisas con esfuerzo e inseguridad.

Introducción

Simon en su libro "Las ciencias de lo artificial" (Simon, 1981) distingue los objetos naturales de los artificiales en función de que estos últimos pueden ser caracterizados atendiendo a funciones y a objetivos. En lo artificial, no sólo es importante saber como son las cosas, sino que es también fundamental caracterizarlos en función de como deberían ser.

Para Simon, a la base de esta distinción se encuentra la diferencia entre las ciencias naturales y las ingenierías. Mientras que las primeras se preocupan de atender a como son las cosas, y, por tanto, dedican especial atención a una postura analítica que descomponga los objetos estudiados, la ingeniería se preocupa de construir cosas, y, por tanto, de sintetizar partes en un todo que sea capaz de cumplir los objetivos[1] propuestos, o que tengan las propiedades requeridas.

Mientras que, tradicionalmente, las disciplinas científicas habían investigado acerca de las ciencias naturales, y las ingenierías acerca de como

---

[1] Aunque no sea el objetivo de esta tesis reflexionar acerca de cuestiones fundamentales en Psicología, es curioso comentar que Simon considera que ésta puede ser considerada como una ciencia artificial al estar centrada en objetos que se hayan limitados básicamente por las condiciones externas, en lugar de las condiciones internas. Es decir, poseen objetivos que intentan alcanzar mediante el uso de estrategias de resolución de problemas. Pinillos (Pinillos, 1981) utiliza argumentos similares en su resumen de las críticas a la explicación causal en Psicología .

eran fabricadas o diseñadas cosas artificiales, en este siglo se ha producido una transformación en el contenido de las ingenierías. Progresivamente, más contenidos científicos, y menos de diseño, han empezado a formar parte de la esencia de sus actividades. La explicación para Simon se encuentra en que, hasta hace poco, el ingenio o "arte", cualidades difícilmente investigables o transmisibles, lo que permitía a los profesionales embarcados en estas tareas llevar a cabo con éxito su labor. Por ello, los académicos, obligados en cierto modo a buscar cuestiones "duras", formalizables, analíticas y enseñables, no seguían este camino con facilidad. Al parecer, anteriormente, todo lo que se sabía acerca del diseño era demasiado "blando", basado en recetas intuitivas e informales, para hallar cabida en el marco más formal de la ciencia pura.

Para Simon, en este siglo se empieza a saber lo suficiente acerca de los principios y las leyes del diseño como para que se pueda empezar a investigar y a enseñar de una manera mucho más adecuada, sobre todo si tenemos en cuenta la importancia que este tipo de conocimientos tiene.

Esta tesis trata acerca del diseño del interfaz entre hombres y máquinas, y es, por tanto, un ejemplo del esfuerzo en la línea de proporcionar conocimientos y metodologías que asistan en la creación de objetos.

El proceso del diseño, según Simon, no es más que un ejemplo de su teoría acerca de la búsqueda de soluciones en medios ambientes complejos. En resumen, su teoría habla de la existencia de múltiples posibilidades a la hora de solucionar un problema que es necesario explorar con el objeto de encontrar la respuesta correcta. Debido a que el número de posibilidades es excesivamente grande para ser exploradas todas individualmente, es necesario fijar criterios para:

1) Determinar criterios para seleccionar una respuesta lo suficientemente buena a pesar de no ser necesariamente la mejor y

2) Guiar la búsqueda de tal modo que un número mínimo de posibles soluciones sean exploradas antes de alcanzar esa alternativa suficientemente buena.

Implicada en el problema de aplicar estos criterios se encuentra la necesidad de poseer un mecanismo de representación del problema que permita realizar estas búsquedas de una manera coherente, lleve registro de las alternativas exploradas y ya desechadas, y además resulte práctico para

encontrar soluciones apropiadas. Por último, una descomposición jerárquica de los problemas, que permitiría tratarlos a distintos niveles de detalle, parece la estructura más adecuada para la mayoría de los sistemas, por lo que el diseño debería guiarse por esta estructura a la hora de construir sistemas complejos.

En nuestra opinión, este esquema es aplicable de una manera bastante general a la disciplina de la interacción hombre-ordenador, la cual, fundamentalmente se preocupa de como diseñar la forma en que seres humanos y máquinas se comunican y se relacionan mutuamente. Aunque un cierto componente de ciencia natural está presente en esta disciplina, y los investigadores en el área se preguntan también acerca de como los usuarios y las máquinas se relacionan, lo cierto es que el carácter de artificial de estos interfaces restan importancia a este aspecto. Lo importante no es cómo son los interfaces, sino cómo deberían ser para cumplir el objetivo de máxima usabilidad.

Para llevar a cabo ese propósito, una gran cantidad de ideas pueden ser traídas a la luz inmediatamente a la hora de realizar el diseño de cualquier interfaz (el espacio de problemas es, sin duda, grande). Estilos de interacción distintos pueden ser considerados (menús, manipulación directa, comandos, etc.), mecanismos de input (ratón, teclado, pantalla táctil, sonido, gestos), control de la interacción (flexibilidad, consistencia, iniciativa por parte del usuario), sistemas de ayuda (on-line, escritos, tutoriales, sistemas inteligentes, insertados en el contexto), público esperado (novatos, expertos, con necesidades especiales), diseño de pantallas (cantidad de información a presentar, agrupamiento de ésta, iluminación, situación de la información), funcionalidad (operaciones que deben ser realizadas por la máquina o por el usuario) y un largo etcétera.

Una forma de manejar esa complejidad es simplemente reducirla.. Por ejemplo, imitar las decisiones tomadas en otros lugares, permitiría evitar la necesidad de contestar a un gran número de preguntas, aunque ello limita seriamente la posibilidad de realizar mejoras a sistemas ya existentes, salvo en aspectos concretos, lo cual disminuiría seriamente las esperanzas de superar lo imitado. También, atender a normas de estilo (guidelines) y standards, tal y como las de ciertos sistemas operativos, permite esa reducción de la complejidad. No obstante, un gran número de decisiones todavía quedarían pendientes a los diseñadores. Aquí, el conocimiento en Psicología Cognitiva aplicada, o mejor todavía, los fragmentos conservados de

ésta en los manuales al uso sobre Interacción Hombre-Ordenador, así como los otros conocimientos de su propia cosecha en ellos contemplados, pueden servir también de guía. Finalmente, pruebas de campo o experimentos controlados pueden servir como últimos criterios de decisión cuando el conocimiento sea débil o inexistente y el presupuesto alto y sin presiones temporales.

Obviamente, crear un buen interfaz es sólo un problema si, como estamos asumiendo, existe algún tipo de criterio que marque la existencia de soluciones mejores o peores. Si no existe ningún criterio de este tipo, entonces el problema desaparece y cualquier solución es aceptable. En un caso menos extremo, el único criterio puede ser el de estabilidad: el programa no deja de funcionar a pesar de que se intenten cosas inesperadas y sin sentido. No obstante, cuestiones como la facilidad de uso, la productividad, la satisfacción o la simple mejora con respecto a sistemas anteriores que realizaban las mismas tareas pueden y deberían ser especificados, de tal modo que los logros alcanzados se hagan evidentes, y se puedan rechazar soluciones que no alcancen las condiciones exigibles.

El mecanismo de representación del problema es, probablemente, la cuestión que esta tesis afronta de manera más central. Por medio de un mecanismo de representación adecuado, el proceso de diseño puede ser realizado sobre un medio genérico: Una representación formal. Los posibles prototipos rechazados son sólo considerados en abstracto, ya que es imposible crearlos todos ellos y simplemente probarlos para ver qué es lo que ocurre, y a partir de ello tomar las decisiones correspondientes. Esto es así incluso en ajedrez, en el que cada prueba sólo necesita el esfuerzo físico de mover una ficha en un tablero, por lo que, en un dominio como el diseño de interfaces, en el que cada modificación necesita un esfuerzo mucho mayor esta situación es indiscutible. Por ello, es necesario algún tipo de representación que permita contemplar el espacio de problemas y sea explorable y modificable hasta llegar a una solución que merezca la pena ser construida para comprobar que tal funciona. Como mucho, un número reducido de soluciones diferentes puede ser analizado con usuarios reales (dos, tres, pero no muchas más) para decidir si la solución hallada es lo suficientemente buena.

Los métodos formales de representación de la interacción son una metodología que permite esto mismo, además de proporcionar mecanismos limitados para evaluar los méritos de las soluciones y de este modo hacer

avanzar el diseño a un costo reducido. Un plano de una casa hace el mismo papel para un arquitecto.

Obviamente, determinar cuál sería el aspecto y la funcionalidad que el interfaz debería tener no es suficiente. Todavía queda el problema de lograr construir el diseño planteado. Una tarea que reviste su propia complejidad, que necesita su propia metodología y que puede llevar el proceso por vías completamente diferentes de las planeadas. A pesar de eso, parece que determinar con claridad el objeto deseado es el mejor camino para hacer que ese proceso de implementación sea lo más simple posible.

Objetivos y Metodología.

El objetivo general de esta tesis es realizar una evaluación de las metodologías basadas en descripciones formales aplicadas a la tarea de realizar predicciones sobre la utilización de interfaces hombre-ordenador por parte de usuarios.

Este objetivo general puede ser desglosado en los siguientes objetivos más concretos.

1) Evaluar qué metodologías permiten tanto la representación del funcionamiento de un interfaz como el análisis de sus características.

Muchas de las notaciones existentes tienden a concentrarse sobre uno de los extremos: representación o análisis. Esta evaluación implica la revisión de las distintas metodologías propuestas y un análisis de sus pretensiones a partir de la literatura existente en relación con ellas.

2) Evaluación de la capacidad de las distintas notaciones para ser utilizadas efectivamente en la descripción de prototipos de diseños.

Ello implica la adquisición y puesta en práctica de la información disponible acerca de como utilizar las distintas notaciones sobre ejemplos al efecto.

3) Evaluación de las predicciones obtenidas a través de las notaciones finalmente seleccionadas en un ejemplo de diseño.

Para ello, se necesitaría analizar el comportamiento de usuarios en una situación de uso y comparar los resultados con el análisis realizado por medio de las notaciones.

La metodología utilizada para llevar a cabo estos objetivos ha sido la siguiente.

1) Revisión de la literatura acerca de las distintas notaciones. Esto se refleja en el capítulo 3 y permite obtener una visión general acerca de las notaciones consideradas.

2) Elaboración de un programa informatico que simularía el comportamiento de ciertas tareas en un ordenado, este programa serviría como ejemplo para el análisis por medio de los distintos métodos formales seleccionados en el último punto.

3) Descripción formal y obtención de predicciones acerca del ejemplo desarrollado a partir de los métodos formales capaces de ofrecer pautas y procedimientos adecuados para lograrlo.

4) Comprobación de la precisión de las predicciones realizadas por los distintos modelos.

5) Analizar la utilidad de las predicciones realizadas por los distintos modelos en relación con el ejemplo considerado.

Estructura de este trabajo

Esta tesis presenta una estructura relativamente simple. El siguiente capítulo está dedicado a exponer cuestiones generales acerca de la disciplina de la Interacción Hombre Ordenador (Human Computer Interaction-HCI). En él se trata en primer lugar la problemática que dio origen a esta disciplina y algunos hechos interesantes en relación con la historia y evolución de los interfaces hombre-ordenador. Pensamos que una estructura que comience analizando el problema que la hizo surgir es seguramente la forma más adecuada de introducir una disciplina aplicada. En este mismo capítulo expondremos las fuentes de conocimiento más básico que utiliza (sobre todo Psicología Cognitiva) y su relevancia e interés. Finalmente, una descripción del proceso de diseño de interfaces, con algunas de las técnicas que la HCI ha desarrollado en relación con él, cierra este capítulo.

El segundo capítulo presenta una introducción acerca de los métodos formales aplicados a la interacción hombre-ordenador.

El tercer capítulo condensa una revisión relativamente amplia acerca de los métodos formales en HCI. Los métodos formales varían ampliamente

en cuanto a su estilo de simbolización: existen notaciones tomadas de la inteligencia artificial y la simulación (reglas de producción), grafos (diagramas de transición), matemáticas y lógica (los modelos del sistema) y otros. Todos esos orígenes no han sido expuestos de una manera minuciosa ya que por sí solos constituyen cuerpos de conocimiento bien establecidos y de considerable extensión.

El capítulo cuarto muestra la aplicación de diversos métodos formales a un ejemplo de programa desarrollado para servir de campo de pruebas de distintas notaciones.

El capitulo quinto muestra una breve introducción a las técnicas de análisis estadístico realizadas en el capítulo 6. Estas técnicas son una variación sobre los análisis de regresión intrasujeto realizados originalmente en estudios similares pero considerando el papel de (Pedhazur, 1982) la autocorrelación. Estos son denominados *pooled time series data* (Green, 1993; Maddala, 1987b; Moore, et al., 1994; Novales Cinca, 1988; Sayrs, 1989), término que hemos traducido por análisis de series temporales ponderadas.

El capítulo Sexto muestra los resultados de un estudio empírico que utiliza tres métodos para analizar un interfaz con objeto de predecir una serie de variables de ejecución. El ejemplo presenta el interés de, aún no siendo un sistema totalmente real, no tener unas condiciones de práctica tan limitantes como otros experimentos hallados en la literatura.

Finalmente, el último capítulo expone las conclusiones alcanzadas acerca del interés que estos modelos nos han demostrado poseer y su posible aplicación en situaciones de diseño.

Capítulo 1

Qué es la Interacción Hombre-Computador

Este capítulo tiene como propósito introducir en primer lugar la problemática que ha generado el desarrollo de la disciplina conocida como Interacción Hombre-Computador (Human-Computer Interaction-HCI), en segundo lugar exponer brevemente su desarrollo a lo largo de sus aproximadamente dos décadas de existencia y en tercer lugar su relación con disciplinas afínes.

1.1. Orígenes de la Interacción Hombre-Ordenador

El diseño de la interacción hombre-ordenador surge como respuesta al problema de la existencia de máquinas o artefactos que son manejados por seres humanos con una cantidad considerable y/o excesiva de esfuerzo, fundamentalmente de tipo cognitivo. Este problema se ha agudizado en las últimas décadas debido a dos procesos concurrentes: la progresiva propagación de estas tecnologías a una población mucho mayor y de carácter menos técnico y la renovación cada vez más rápida de la tecnología lo cual obliga a estos usuarios aaprender nuevos métodos y sistemas de manera contínua.

Resulta interesante señalar una explicación económica del problema. Hasta hace unas décadas, los ordenadores eran lo suficientemente caros como para que el salario de la gente que los utilizaba significara una parte sustancial del costo de una transacción con éstos. Por ello, el aspecto a primar era la velocidad de ejecución de la máquina aunque ello supusiera obligar al usuario a necesitar mayor formación, memorizar más comandos o ser más preciso en la sintáxis utilizada.

En la actualidad, la relación se ha invertido, y existen máquinas que tienen un precio considerablemente inferior al de los salarios de los usuarios que las manejan. Por ello, lo importante hoy en dia es que los usuarios realicen mejor y más rápidamente sus tareas, no teniendo importancia que las máquinas pierdan la gran eficiencia y velocidad que podrían alcanzar con interfaces tan simples como los anteriormente existentes.

Este esfuerzo por facilitar las tareas de los usuarios ha producido resultados realmente notables. Probablemente, el más espectacular ha sido la trasformación del estilo de interfaz basado en comandos a los interfaces gráficos (Graphical User Interfaces-GUI) los cuales se han convertido en totalmente comunes entre los usuarios de ordenadores personales y estaciones de trabajo y que parecen destinados a continuar como predominantes durante los próximos años a pesar de que otros estilos (p.e. realidad virtual) apuntan perspectivas futuras de gran interés.

Este nuevo estilo de interacción, como puede suponerse, no ha supuesto la eliminación completa de interfaces hombre-máquina innecesariamente difíciles de usar o simplemente mejorables. Un ejemplo tomado de un manual al uso, cuyos autores son provinientes a partes iguales del campo de la Ciencia de los Computadores y la Psicología, (Dix, et al., 1993) permite ver como incluso uno de los interfaces más sencillos posibles en apariencia puede dar lugar a problemas:

"…los primeros cajeros automáticos daban a los clientes el dinero antes de devolver las tarjetas. Desgraciadamente, ello llevó a que muchos clientes se dejaran las tarjetas en ellos. Esto ocurría a pesar de mensajes en la pantalla indicándoles que debían esperar para recoger las tarjetas. Este problema es conocido con el nombre de "problema de finalización". El objetivo principal del cliente es conseguir el dinero; cuando este objetivo está logrado, el usario no completa o finaliza las subtareas que han quedado pendientes (…).

(…). Los bancos (al menos algunos de ellos) pronto cambiaron el orden del diálogo de modo que la tarjeta se recupere antes de que se retire el dinero."

Posiblemente este ejemplo sea demasiado extremo y el siguiente párrafo, tomado de Smith and Mossier (1986) (Smith y Mosier, 1986) situe las cosas en una orbita más apropiada

"Probablemente, no habrá muchos casos en los que un único defecto de programación será el culpable del mal funcionamiento del sistema ya que el ser humano siempre es capaz de adaptar su conducta para compensar esta situación. No obstante, cuando una equivocación se acumula a otra, existe un límite a como los sujetos pueden adaptarse a un sistema mal diseñado. Cuando el sistema puede ser utilizado opcionalmente, la mala calidad puede apreciarse en que es infrautilizado. Cuando debe obligatoriamente ser utilizado, el fallo puede apreciarse en tiempos de ejecución lentos y en falta de satisfacción del usuario."

Ante la existencia de este tipo de problemas, los encargados de la construcción de interfaces hombre-computador pueden adoptar diversas actitudes que darán una solución más o menos completa a todos ellos. De una manera amplia, Moran (Moran, 1981a) describió tres posibles actitudes y las consecuencias que es posible esperar en cuanto a su efecto sobre los interfaces. Estas son:

• La actitud de los "tecnologos". Para ellos, la solución de los problemas vendrá dada por la aparición de tecnología cada vez más poderosa, más rápida y más "grande", la cual será capaz de cambiar cualitativamente la forma en que la gente realiza sus tareas. Por ejemplo, un representante de este estilo de pensamiento opinará que los monitores de alta resolución han hecho más por la mejora de la interacción que cualquier cantidad de "ingeniería humana" que se pudiera aplicar sobre los anteriormente existentes, basados en caracteres.

Ahora bien, aunque es indudable que las mejoras tecnológicas introducen grandes avances en la forma en que los usuarios interactuan con las máquinas, ello no impide que, dentro de lo realizado mediante un tipo de tecnología, se produzcan grandes diferencias que vienen a demostrar la importancia de los análisis cuidadosos de las posibilidades que ésta ofrece.

• La actitud de los "diseñadores": Para los diseñadores, la mejor forma de dar solución a los problemas de los usuarios es simplemente prestar más

atención a 'estos. El diseñador es, después de todo, también un usuario y, por tanto, puede tener las intuiciones adecuadas para predecir qué será fácil o difícil para los otros usuarios.

La limitación de esta aproximación es obvia. Las intuiciones de los diseñadores no necesariamente se ajustan a las de los usuarios. El diseñador, apoyándose en una psicología popular no tiene forma de respaldar sus intuiciones, y las intuiciones acerca de la conducta psicológica compleja (incluso acerca de la de uno mismo) pueden ser notablemente decepcionantes.

Este fenómeno es fácilmente apreciable en los análisis informales que se realizan a menudo cuando se desarrollan nuevos sistemas. Cada usuario informal proporciona un nuevo bloque de aparentes dificultades o sugerencias que el desarrollador podía o no haber detectado, pero que no sabe hasta qué punto debería considerar como importantes o no y que, en cualquier caso, probablemente no justificarán el retraso del proyecto actual.

• La actitud del "psicólogo" es que la conducta del usuario debe ser examinada rigurosamente: el usuario tiene que ser observado objetivamente, de modo que sea efectivamente comprendido y el conocimiento así obtenido sea empaquetado en una psicología aplicada del usuario que diseñe sistemas que se ajusten a sus necesidades.

Esta clasificación en actitudes no obstante no debería ser vista como una propuesta en que las últimas desplazarían a las primeras dejándolas sin aplicación o contenido. Cada una de ellas es importante dentro de un momento del desarrollo de un producto y existen aspectos que sería innecesario tratar rigurosamente cuando pueden ser previamente detectados y corregidos[1] de modo más simple.

Ahora bien, para que la tarea de introducción de consideraciones acerca de la psicología de los usuarios sea posible es necesario proporcionar las

[1] A pesar de la distinción que hemos establecido entre las tres actitudes, no quisieramos que esta terminología se convierta en algo así como la definición de roles profesionales dentro de un equipo de trabajo siendo el diseñador del interfaz algo fundamentalmente distinto del psicólogo. En el resto del texto preferiremos utilizar una terminología en la que "diseñador" o "diseñador del interfaz" sea la persona encargada de esta tarea, la cual, usualmente, tendría una cantidad variable de conocimientos acerca de psicología aplicada a los interfaces, y, por tanto, actuaría en función de la situación concreta a manejar utilizando su propia psicología popular o realizando observaciones más formales de la ejecución del sistema con usuarios reales.

bases que respalden sus actuaciones. Un problema fundamental aquí es la dificultad que supone corregir programas una vez realizados para atender evaluaciones post hoc que indican deficiencias en sistemas que, por otro lado, funcionan correctamente desde el punto de vista del programador.

El modelo Seheim (Pfaff, 1985) supuso un primer avance en la dirección de facilitar las posibles correcciones al postular la necesidad de la separación de las capas de funcionalidad e interfaz con el usuario de modo semejante al que en las bases de datos se produce al distinguir entre datos y forma de acceder o utilizarlos. De ese modo, se garantiza la posibilidad de realizar modificaciones o mejoras en una de las capas sin ser necesario modificar las otras.

No obstante, si tenemos en cuenta el esfuerzo que implican los interfaces de usuario dentro de la labor de programación total, esta posibilidad deja de ser suficiente para garantizar que las mejoras serán realmente introducidas. Por ejemplo, Smith y Mossier (1986) hablan de un 30 a un 35 por ciento de código estuvo dedicado al interfaz de usuario, mientras que Browne (1994) comenta cifras más recientes que aumentan la estimación al 47-60%. Es poco probable que esta cantidad de esfuerzo sea dejado de lado aun a pesar de encontrar evaluaciones negativas por parte de las pruebas hechas con usuarios reales. Muy probablemente, las únicas modificaciones que presentan esperanzas de ser aceptadas son aquellas que no supongan cambios radicales.

En definitiva, un modelo de diseño basado en la elaboración de prototipos, su posterior prueba y modificación y construcción de un nuevo prototipo en un ciclo iterativo que terminaría cuando el producto alcanzara una calidad considerada aceptable es, en muchos casos, prohibitivo debido a su excesivo costo. Debido a que el esfuerzo realizado es tan grande, existirá una enorme presión en dirección a no realizar ninguna modificación o en limitarse únicamente a aquellas que no necesiten un esfuerzo extremado para su consecución, independientemente de lo graves que sean los defectos desde el punto de vista de los usuarios.

Debido a lo expuesto en el párrafo anterior, Newell y Card (Newell y Card, 1985) postularon el siguiente principio: el diseño está donde está la acción, no en la evaluación. Para que la psicología pueda influir en el diseño es necesario que exista una participación previa en éste que, desde un principio, incorpore las consideraciones necesarias acerca del usuario. En caso

contrario, a pesar de las evidencias en cuanto a usuarios teniendo problemas con el interfaz, resulta difícil pensar en que realmente se realizarán modificaciones al trabajo realizado.

Esta consideración introduce la necesidad de los métodos formales de descripción y evaluación. Estos métodos tienen como objeto realizar especificaciones que señalen como serán los artefactos que serán construidos, y, en caso necesario, permiten realizar análisis y diagnósticos que permitan realizar modificaciones y mejoras previamente a la existencia de prototipos. No entraremos en una descripción profunda de este aspecto todavía, ya que lo retomaremos en secciones siguientes con mayor profundidad. Sólo señalar de nuevo que la promesa ofrecida por los métodos formales es la de determinar las cuestiones de usabilidad previamente al desarrollo de los productos.

1.2. Historia de la interacción hombre-ordenador

En la tabla 1.1 se muestra una comparación entre la evolución del hardware y el software y los avances realizados en el campo específico de la interacción hombre-ordenador.

	Hardware/software	Estado de la HCI.
0 1940-47	"**Arriba y Abajo**" De los transistores a los tubos de vacio	**El diseñador como usuario** El diseñador es el propio usuario
1 1948-55	**Primeros éxitos** Tubos, registros magnéticos.	**La máquina domina** La gente se adapta a la máquina
2 1956-63	**Traficantes de papel** Transistores y almacenes centrales, programas de control I/O. Batch, execs, Petri nets *Communications of the ACM*	**Ergonomía** Ergonomía de consola. Lenguajes de control. Simuladores, gráficos. *Avance hacia la HCI* *Sketchpad* *Se inventa el ratón*

3 1964-71	**Comunicadores** Terminales interactivos	**Estudios hombre-maquina. Interactividad** Servicios tiempo-compartido. Terminales interactivos. Síntesis lenguaje. *International Journal of Man-Machines Studies*
4 1972-79	**Recursos personales** Ordenadores personales, superordenadores, VLSI, almacenamiento masivo grande, bancos de datos, video texto	**Reglas de diseño HCI** Ordenadores personales. Reglas de diálogo. *Visicalc.* *Altair and Apple, PCs*
5 1980-87	**Ayudas a la acción** Ordenadores personales con potencia y capacidad de almacenamiento de mainframes más gráficos y procesamiento del lenguaje	**Principios sistémicos naturales al usuario** *Xerox Star, IBM PC, AppleMacintosh,*
6 1988-93	**Simbiosis hombre-máquina** Almacenaje y lógica óptica	**Diseño automatizado similar al usuario** Sistemas multimodales integrados.

Tabla 1.1: Generaciones en el desarrollo del hardware-software y avances que han cursado paralelamente a éstos desde el punto de vista de la interacción hombre-ordenador (adaptado de (Gaines y Shaw, 1986)).

Como es posible ver, no es hasta la tercera generación (notar que hay una generación cero) que podemos hablar de una verdadera atención a las cuestiones relacionadas con la interacción hombre-ordenador. Hasta entonces, se había pasado por una fase inicial en que el diseñador era el usuario fundamental de sus propios programas y, por tanto, no existía una preocupación por facilitar la tarea a otros usuarios potenciales, y, por otra en la que el tiempo de la máquina era mucho más caro que el de sus usuarios por lo que éstos eran obligados a adaptarse a las exigencias de aquella. La segunda generación empieza a considerar al usuario, pero de momento las consideraciones ergonómicas prevalecen sobre las cognitivas o de diálogo, aunque los avances realizados aquí ponen la base para el avance

(*breakthrough*) de la interacción hombre-ordenador, que se produce fundamentalmente durante la tercera generación y continua vigorosamente durante la cuarta. En el año 1965 cuando se organiza la primera conferencia sobre HCI (*IBM Scientific Computing Symposium on Man-Machine Communication)* y una serie de comentarios como el siguiente significan el reconocimiento del potencial interés y la necesidad de avance en esa dirección:

"Se aproxima rápidamente un futuro en el que los programadores profesionales se contarán entre los menos numerosos y los menos significativos usuarios de los sistemas" (Mills, 1967 recogido en Gaines & Shaw, 1986).

En la cuarta generación el crecimiento empieza a aumentar exponencialmente y, entre otros acontecimientos, se lleva a cabo la investigación que daría lugar al libro "The Psychology of Human-Computer Interaction" por Card, Moran y Newell, probablemente la primera aportación madura al campo. En él se pusieron las bases para buena parte de la investigación de los años 80, sugiriendo una de las tendencias o "modos de hacer"(v. secciones siguientes) más influyentes en la actualidad. Este trabajo se desarrolló en el Xerox Palo Alto Research Center (PARC), que desde los años 70 había empezado una línea de investigación preocupada por la automatización de sistemas de oficina y que llevó a avances en ordenadores personales interactivos con pantallas gráficas de alta calidad (El Xerox Alto fue el primero de ellos) y conectados por la red Ethernet.

El ambiente en el centro de Palo Alto es descrito como muy atractivo por Heid y Norton (Heid y Norton, 1989) ya que:

"…Xerox formó el PARC en 1970, dando rienda suelta para llevar a cabo investigación y desarrollo, sin obligación de producir productos comerciales. PARC combinó una atmósfera académica con los bolsillos profundos de una gran compañía-una combinación que atrajo a los mejores científicos del momento."

Ese ambiente tan interesante dio lugar desde el punto de vista del avance de la HCI a un producto, el Xerox Star, que puede considerarse como el primer computador que incorporó conscientemente cuestiones acerca del diseño de la interacción, que realizó prototipos previos y que, en general, propuso principios que han perdurado en muchos sistemas actuales. Por

ejemplo, (Smith, et al., 1982) afirman haber perseguido los siguientes principios al diseñar el Star:

• Modelo conceptual familiar al usuario. Un modelo conceptual es una representación externa del comportamiento de un dispositivo. Esta representación externa está en muchos casos diseñada para ocultar el funcionamiento interno del dispositivo, de tal modo que el usuario pueda manejarlo sin conocer aquél. En el caso del Xerox Star, el modelo conceptual fue el entorno de una oficina en la que, por ejemplo, los archivos de ordenador eran representados por medio de iconos que asemejaban hojas de papel y los directorios como archivadores o carpetas.

• Ver y señalar frente a recordar y teclear. Los comandos recordados y tecleados son más rápidos de ejecutar que, por ejemplo, los menús, cuando el usuario adquiere un grado de experiencia grande con un programa. No obstante, la inmensa mayoría de usuarios actualmente son novatos y casuales, y prefieren ver las opciones que tienen disponibles y elegirlas de menús, etc.

• Lo que ves es lo que consigues (what you see is what you get): En este momento los tipos de letras, los gráficos etc. empiezan a ser visibles en la pantalla, de modo que el usuario tiene menor necesidad de imprimir para saber el aspecto que va a tener aquello que está haciendo.

• Comandos universales. Hay ciertas funciones que pueden considerarse universales: borrar, mover, salir, etc. Estas funciones pueden homogeneizarse entre programas de tal modo que el usuario no necesite variar de forma de ejecución entre ellos.

• Consistencia. La consistencia hace referencia a la posibilidad de realizar funciones relacionadas con acciones relacionadas. Este es un concepto bastante amplio que abordaremos posteriormente.

• Simplicidad. Las acciones más usuales deberían hacerse con mucha facilidad, aunque ello signifique que las menos usuales se tengan que hacer de formas un poco más complejas.

• Interacción sin modos. Un modo es un estado de sistema en que las ordenes o comandos enviados al ordenador son interpretados de un modo diferente al usual. En una hoja de cálculo concreta, apretar la tecla de la flecha hacia la derecha puede significar sumar una celda o mover un

caracter, dependiendo de si lo que se está introduciendo es una fórmula o un texto. Este modo es invisible al usuario y muy a menudo se producen confusiones en el uso de estas teclas.

• Personalización (*Customization*) al usuario: Los usuarios poseen preferencias o formas de trabajo distintas que los programas pueden reconocer y utilizar.

Un concepto que ellos no señalan y que también incorporó el Xerox Star es el de "primero objeto, luego comando", más adecuado para un lenguaje de manipulación directa, frente al principio de "primero comando, luego objeto" más habitual en lenguajes basados en comandos.

Xerox no aprovechó comercialmente todo ese esfuerzo investigador, por lo que tuvo que ser Apple la empresa que, primero con el Lisa y már tarde con el Macintosh, logró que las ventajas de estos principios alcanzaran al público en general.

En la actualidad, el paradigma propugnado en PARC se ha convertido en el más común entre la mayoría de los sistemas informáticos y, por tanto, es el momento en que nuevos avances empiezan a germinar. Interfaces inteligentes, realidad virtual, interfaces multimodales, son algunas de las propuestas sobre las cuales podrían girar la forma de comunicarse con ordenadores en un futuro cercano.

En la actualidad, HCI se ha convertido en una disciplina relativamente estable: gran número de compañías mantienen laboratorios en los que se ayuda a desarrollar y evaluar software y hardware tal y como muestra un número reciente de *Behaviour and Information Technology*. La *Association for the Computing Machinery* (ACM) tiene un grupo de interés especial dedicado al área (SIGCHI) que publica un boletín trimestral. Además, un congreso de gran tamaño (CHI) se celebra anualmente así como una gran cantidad de otros de menor tamaño.

1.3. Definición de Interacción Hombre Computador y relación con otras areas de la Psicología

En esta sección presentaremos algunas de las definiciones de HCI realizada en manuales al uso, así como unas notas acerca del tipo de conocimientos que implica. Por último, mostraremos qué diferencias existen

entre este campo y otros relacionados como son la Ergonomía y los Factores Humanos.

1.3.1. Definiciones de Interacción Hombre-Ordenador

Una definición interesante es la de Johnson (1992), el cual considera la HCI *"…como el estudio de la interacción entre gente, ordenadores y tareas. Esto está principalmente relacionado con la comprensión de como la gente y los ordenadores pueden interactivamente realizar tareas, y como tales sistemas interactivos son diseñados."*

En cuanto a los conocimientos implicados en esta disciplina el siguiente párrafo (Dix, et al., 1993) nos parece lo suficientemente revelador:

"HCI es indudablemente una materia interdisciplinar. El diseñador ideal de un sistema interactivo debería tener experiencia en una amplia serie de tópicos: psicologia y ciencia cognitiva que le proporcionen conocimiento de las capacidades perceptivas, cognitivas y de solución de problemas; ergonomía para las cuestiones físicas; sociología para ayudarle a entender el contexto más amplio de la interacción; ciencias de la computación e ingeniería para permitirle construir la tecnología necesaria; negocios, para ser capaz de vender; diseño gráfico para producir una presentación del interfaz efectiva; escritura técnica para escribir los manuales, y la cosa continua. Esto es obviamente demasiada experiencia para ser poseida por una sola persona (o cuatro!), quizás incluso demasiada para el equipo de diseño promedio. Verdaderamente, aunque la HCI es reconocida como una materia interdisciplinar, en la práctica la gente tiende a adoptar una postura en una u otra dirección. No obstante, no es posible construir sistemas interactivos efectivos desde una disciplina en solitario. Información es necesaria desde todos los lados. Por ejemplo, un diseño gráfico elegantemente diseñado puede ser inutilizable si ignora las constricciones del diálogo o las limitaciones psicológicas del usuario."

En lo anterior, y a pesar de existir etiquetas para el campo un tanto diferentes, tal y como, por ejemplo, Ingeniería de sistemas cognitivos (Woods y Roth, 1988), Ingeniería Cognitiva (Norman, 1986; Norman, 1987), Psicologia del Software (Shneiderman, 1986), Ergonomía Cognitiva (Long y Whitefield, 1989), podría decirse que existe un consenso relativamente amplio en cuanto a las cuestiones anteriormente señaladas por lo que no ampliaremos esta sección. De mayor interés es, en nuestra opinión, la siguiente sección, dedicada a diferenciar entre distintas subareas de

investigación aplicada que, aparentemente, comparten la misma área de acción.

1.3.2. La distinción entre areas de la Psicología aplicada relacionadas con el estudio de máquinas y seres humanos

Donald Norman se pregunta a sí mismo al introducir el término de ingeniería cognitiva:

"¿Por qué un nuevo término? Después de todo, ya existen muchos términos utilizados para describir el lado aplicado de la Psicología: ergonomía, factores humanos e ingeniería psicológica. En mi opinión, creo que se necesita un nuevo término porque se necesita una nueva aproximación."(Norman, 1987).

Efectivamente, los términos de ergonomía y factores humanos sobre todo, parecen solaparse de alguna manera con el campo de la HCI. En la sección siguiente veremos una serie de opiniones acerca de cómo distintos autores han conceptualizado las relaciones entre estas disciplinas, así como las fuentes más básicas de las que toman información. Además mostraremos los resultados de un trabajo presentado en otro sítio (Valero Mora, et al., 1994) que supone una aproximación empírica basada en la bibliografía al problema de la clasificación de disciplinas.

Para Barber y Laws (1989) la ergonomía cognitiva está concebida "debilmente" como un subconjunto de la disciplina de la ergonomía. Así, la ergonomía cognitiva estaría relacionada con la Psicología cognitiva y la Ciencia Cognitiva. Una definición rápida de la función de la ergonomía cognitiva es que trata del trabajo mental hecho con ayuda o en el contexto de los ordenadores. En ese sentido puede ser considerada como una contribución parcial al área más general de la HCI (Human-Computer Interaction).

Dix et. al. (Dix, et al., 1993) señalan "(...)Tradicionalmente, los ergonomistas han estado interesados fundamentalmente en las características físicas de las máquinas y los sistemas, y en como estas afectan la ejecución del usuario. Los Factores Humanos incorporan estas cuestiones, y también cuestiones más cognitivas. Los términos son usados a menudo intercambiablemente, siendo Ergonomía la preferida en Gran Bretaña y Factores Humanos en las partes anglo-parlantes de Norte America".

Johnson (1992), conecta el campo de la HCI con otras areas, haciendo los siguientes comentarios. En primer lugar, con respecto a los Factores Humanos y la Ergonomía dice:

"Factores Humanos es uno de los variados términos utilizados para describir el estudio de la gente y su conducta en el contexto de utilizar máquinas, herramientas y otros desarrollos tecnológicos en el contexto del manejo de máquinas. El término apareció en Estados Unidos: En Gran Bretaña, las mismas áreas de interés son contempladas en el estudio de la Ergonomía. No obstante, en Estados Unidos, la Ergonomía está definida mucho más estrechamente, y está principalmente centrada en asegurar que el diseño de una máquina tenga en cuenta las características físicas de la población de usuarios pretendida. Por ejemplo, los requisitos de tamaño y fuerza de una tecla en un teclado son comprobados para asegurarse de que están dentro del rango óptimo para permitir a una población de usuarios con tamaños y potencia de dedos bien conocidas oprimir la tecla fácil y cómodamente.(...) La Interacción Hombre Ordenador incluye partes apropiadas de Factores Humanos y Ergonomía. Está interesada en la comprensión de como los ordenadores y la gente pueden interactivamente realizar tanto tareas nuevas como tareas existentes. HCI está interesada en todos los aspectos del diseño y uso de ordenadores".

Y, en segundo lugar, con respecto a la Psicología, dice:

"La teoría psicológica y sus métodos forman una contribución principal a nuestra comprensión de la HCI. (...) Es difícil decir qué areas de la Psicología son las más relevantes a la HCI puesto que todos los aspectos de la conducta humana pueden, en la ocasión apropiada, tener algún efecto sobre la interacción de la persona con un ordenador y un ordenador puede afectar la conducta humana de muchas maneras".

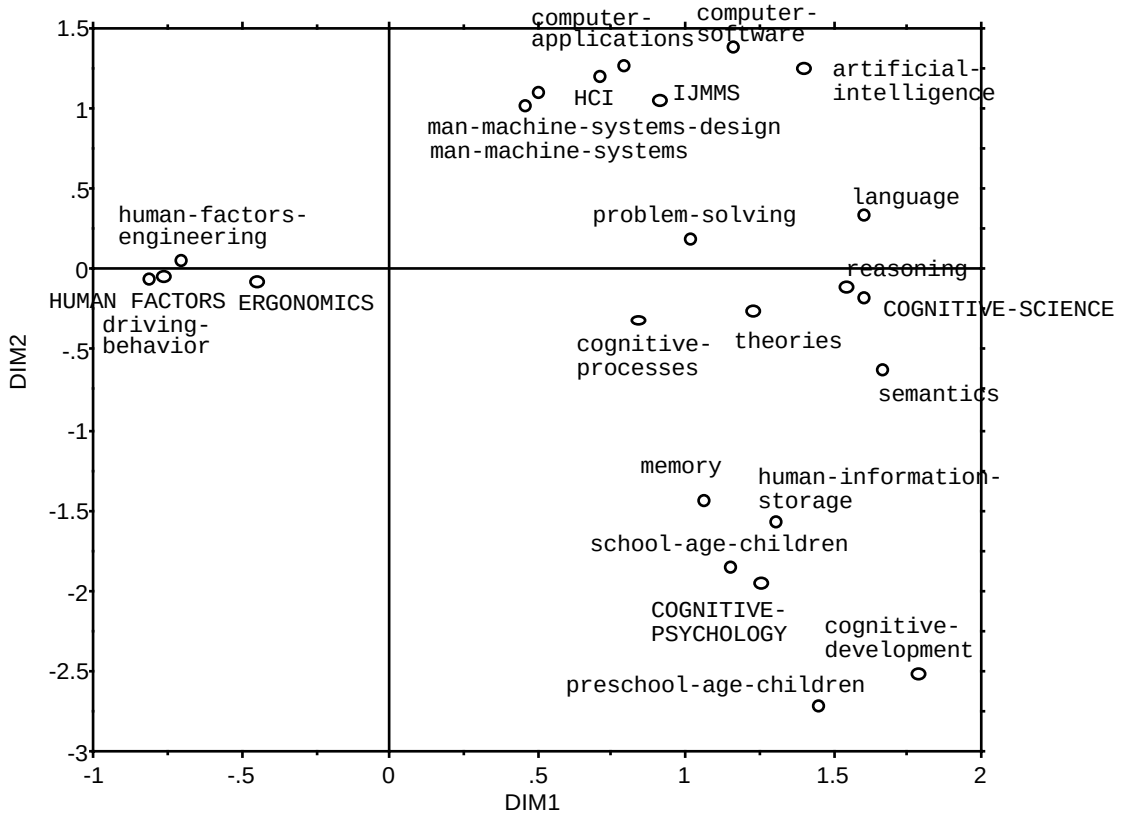
Esta interrelación entre distintas disciplinas y la diferenciación entre ellas fue puesta a prueba mediante un análisis bibliométrico de los descriptores utilizados por la base de datos Psyclit para etiquetar los artículos aparecidos en seis diferentes revistas con títulos que incluían términos similares a los arriba definidos. Estas revistas fueron las mostradas en la tabla 1.2.

La matriz de tabulaciones cruzadas entre revistas y palabras claves fue sometida a un análisis factorial de correspondencias (Gifi,

Revistas	Adscripción area	Número de palabras clave
COGNITIVE PSYCHOLOGY	P. Cognitiva	1731
COGNITIVE SCIENCE	C. Cognitiva	879
ERGONOMICS	Ergonomía-HF	5889
HUMAN COMPUTER INTERACTION	HCI	596
HUMAN-FACTORS	Ergonomía-HF	4972
IJMMS	HCI	3159

Tabla 1.2. Revistas utilizadas.

1990,Greenacre, 1989) que dio como resultado principal las gráficas de la figura 1.1 y las inercias y proporciones de varianza explicadas de la tabla 1.3. Las palabras indicadas en el gráfico fueron seleccionadas a partir de un método gráfico que permitió seleccionar aquellas con altas contribuciones a las dimensiones además de ser bien explicadas por ellas (v. Valero et. al., 1994 para detalles).



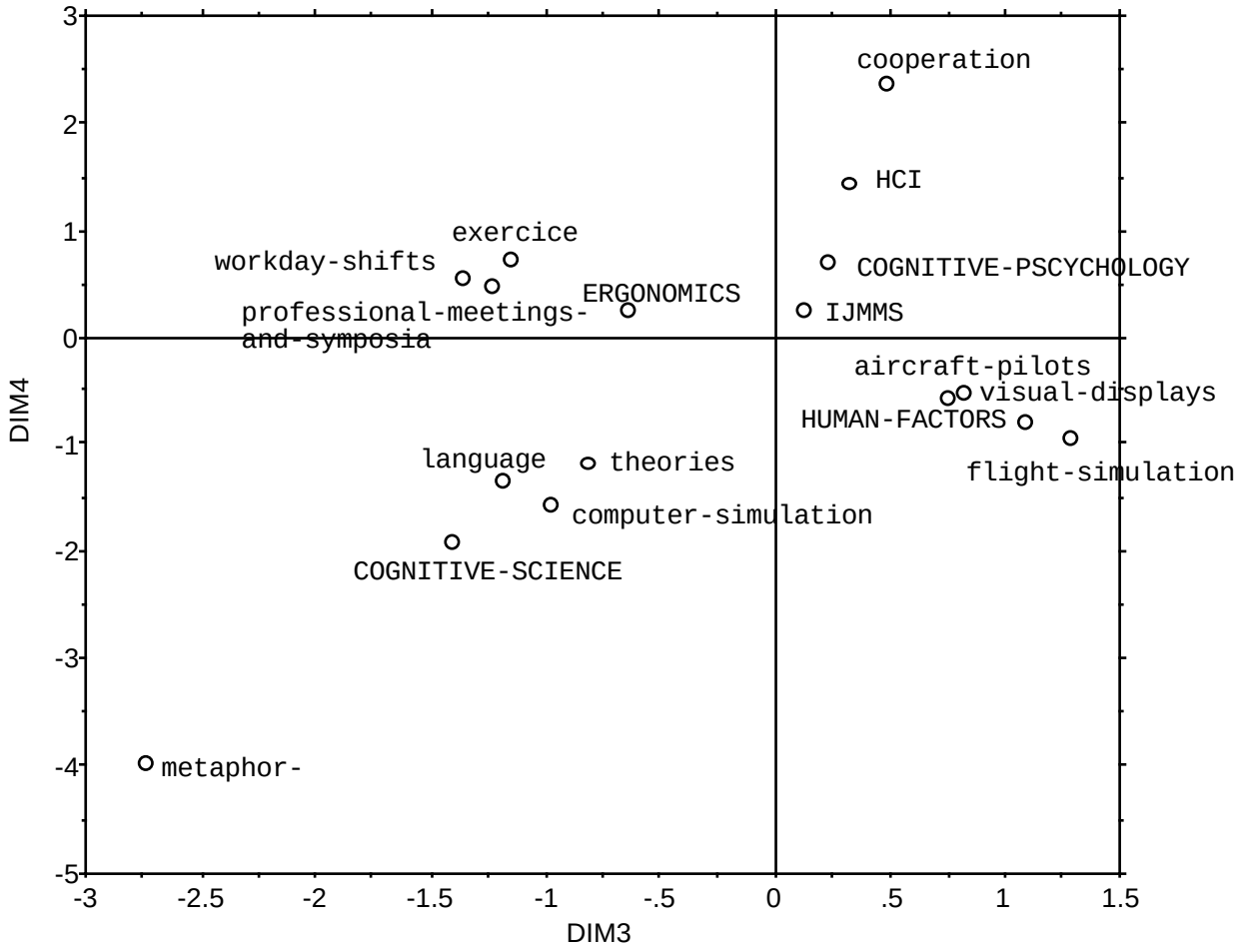


Figura 1.1: Mapa de palabras claves y revistas analizadas.

DIMENSION	VALOR SINGULAR	INERCIA	PROPORCION EXPLICADA	PROPORCION ACUMULADA
1	0.694	0.481	0.357	0.357
2	0.596	0.355	0.263	0.620
3	0.446	0.199	0.148	0.767
4	0.414	0.171	0.127	0.894
5	0.378	0.143	0.106	1.000
TOTAL		1.350	1.000	1.000

Tabla 1.3: Inercias y proporciones de inercia explicadas por las dimensiones encontradas por medio del análisis de correspondencias.

Como es posible ver, las revistas de Ergonomía y Factores Humanos tienden a agruparse conjuntamente en el gráfico de la primera y segunda dimensión, mientras que las revistas dedicadas a HCI hacen lo mismo. Cognitive Science parece estar bastante próxima a estas últimas habiendo términos como lenguaje, solución de problemas e inteligencia artificial

compartidos entre ambas. Cognitive Psychology es una revista más general en comparación con las otras cinco. La gráfica de las dimensiones tres y cuatro ilumina diferencias menores entre Ergonomics y Human Factores en donde se muestra la predilección de esta última por el área de diseño de cabinas de pilotos.

Esta última área parece más cercana a la revista Cognitive Science habiendo términos como solución de problemas, lenguaje e inteligencia artificial que parecen bastante cercanos. Cognitive Psycology en cambio parece claramente diferenciada en el gráfico

En resumen, este estudio parece demostrar la relativa consistencia y diferenciación del campo de la Interacción Hombre-Ordenador de otras areas básicas y aplicadas con ella relacionada. Pasaremos ahora a comentar la forma en que esta disciplina puede proporcionar conocimientos útiles para diseño de sistemas hombre-máquina y qué fuentes puede utilizar para alcanzar este propósito.

1.4. Líneas de pensamiento e investigación en interacción hombre-ordenador

En este apartado revisaremos las aportaciones de la Psicología Cognitiva al estudio de la Interacción Hombre-Computador y los enfoques teóricos existentes dentro de la disciplina.

1.4.1. Aportaciones de la Psicología Cognitiva al estudio de la Interacción hombre-computador

A pesar de que, aparentemente, la Psicología Cognitiva parece una disciplina que debería poseer una cantidad de conocimientos y recursos listos para ser aplicados en un dominio como el que estamos considerando, lo cierto es que a lo largo de la evolución de la disciplina progresivamente se ha hecho evidente las limitaciones de esta aplicación tan directa y la necesidad de replantear la cuestión para realmente proporcionar recursos útiles a la situación.

Esta situación está reflejada por Landauer (Landauer, 1987), el cual ofrece una discusión acerca de las tres maneras en que, en su opinión, podría aplicarse la Psicología Cognitiva a la HCI.

1) Aprovechando los conocimientos básicos que la Psicología Cognitiva ya posee y aplicándolos a las situaciones de diseño

La Psicología Cognitiva posee una gran cantidad de información acerca de tiempos, duraciones, intensidades etc. en relación con procesos como la memoria, la percepción, etc. ¿No podría aprovecharse esos conocimientos para diseñar herramientas que fueran más faciles de usar? Landauer opina que no, y pone un ejemplo basado en la memorabilidad de distintas estructuras de lenguajes de comandos que demuestran lo difícil que es aprovechar los datos recogidos con objetivos científicos a situaciones aplicadas.

El problema fundamental para él es uno de robustez, el cual es similar aunque no completamente al de generalizabilidad. La cuestión está en que un determinado efecto puede observarse en una situación controlada en la que otros factores hayan sido eliminados, pero ese mismo efecto puede ser completamente diluido en una situación práctica por una variedad de elementos. Así, puede ser más interesante detectar esos elementos accidentales antes que preocuparse por principios téoricos que difícilmente tendrán algún efecto (a pesar de ser científicamente ciertos).

En la investigación científica, realizar una división del problema puede ser la estrategia adecuada. No obstante, es necesario realizar estudios que muestren como unir posteriormente los conocimientos en organizaciones con sentido.

2) Aplicando el estilo de crear teoría que la Psicología Cognitiva tiene aunque sin utilizar los conocimientos concretos

Landauer se está refiriendo en este caso a la utilización de modelos de ingeniería aproximados, una perspectiva propuesta por Card, Moran y Newell (Card, et al., 1983) y que sigue teniendo una gran influencia en la actualidad. Estos modelos suponen una aplicación simplificada de los conocimientos que la ciencia básica posee a situaciones concretas, de tal modo que se puedan extraer impresiones aproximadas acerca de las cuestiones de relevancia. El modelo de Card, Moran y Newell es expuesto con más profundidad en capítulos posteriores.

Landauer se muestra relativamente crítico con esta perspectiva basándose en dos puntos:

• Los modelos de ingeniería parecen demasiado centrados en la evaluación de productos y no son capaces de ofrecer conocimientos que influyan sobre el proceso de diseño.

• Hasta ahora no parecen haber generado modelos que proporcionen una adecuada explicación de *todos* los factores importantes en la evaluación de un diseño, ya que al ser modelos pseudoteóricos se centran en procesos concretos y dejan muchos otros sin considerar.

En nuestra opinión, las dos críticas anteriores no deberían ser consideradas muy negativamente. Tener herramientas de evaluación limitadas fue el propósito original de Card et. al y continua siendo un objetivo lo suficientemente valioso como para merecer la pena el esfuerzo.

3) Emplear los métodos analíticos y de investigación

Hay dos hechos metodológicos muy simples y elementales pero fundamentales que son dados por hecho por los psicólogos experimentales y que muy a menudo muchos otros profesionales no son capaces de percibir. Estos son 1) que la conducta es muy variable entre sujetos y entre momentos y 2) que sí es posible recoger datos objetivos acerca de la conducta.

Los psicólogos pueden realizar contribuciones valiosas simplemente insistiendo en observar suficiente número de usuarios sobre suficiente número de tareas representativas y tomando medidas sistemáticas sobre tiempo en la tarea, número de errores, opiniones, ejemplos de tipos de errores e intuiciones del usuario y el diseñador acerca de posibles mejoras a realizar. Los diseñadores se sorprenden habitualmente del interés de estas observaciones. Lo que podría ser denominado como la falacia de la intuición ingenua (que todo el mundo responderá o actuará similarmente a uno mismo) parece estar muy amplia y profundamente extendida.

Por otro lado, hay que señalar que los métodos habituales en Psicología experimental, los cuales buscan sobre todo localizar variables que tengan un efecto claro una vez eliminadas una serie de otras variables presentes en la situación, pueden no ser totalmente apropiadas para la HCI. Por el contrario, paradigmas de investigación exploratoria antes que de prueba de hipótesis pueden parecer más interesantes. Realizar experimentos puede ser la peor manera de investigar en el campo debido a que la elección de variables, factores y niveles está condenada a estar pobremente motivada. Monk y Whright (Monk y Whright, 1991) señalan un problema añadido referido a la selección de usuarios para realizar experimentos en HCI. Mientras que en investigación pura no resulta excesivamente problemático utilizar

estudiantes pregraduados, hacer lo mismo en un estudio de un sistema computerizado puede no ser apropiado ya que la ejecución en las tareas implicadas puede ser críticamente dependiente de la experiencia y conocimientos previos de los sujetos estudiados. La investigación aplicada en definitiva parece necesitar una mayor ponderación de los parámetros de validez ecológica respecto a los de validez interna.

1.4.2. Enfoques teóricos en Interacción Hombre-Computador

Como es posible ver en la sección anterior, la HCI parece obligada a generar de algún modo los conocimientos y técnicas apropiados para llevar a cabo su tarea, asistir en el diseño de interfaces con objeto de mejorarlos. Estos conocimientos y técnicas han sido hasta ahora según Lewis (1990) producidos según cinco enfoques o estilos que expondremos a continuación:

1. La ciencia psicológica acumulativa es el objetivo

Este enfoque propugna la necesidad de generar conocimientos generalizables y teóricos que los diseñadores puedan utilizar a la hora de realizar su trabajo. Este estilo tiene como representantes principales a Card, Moran y Newell (Card, et al., 1983) los cuales propusieron un estilo de ciencia basado en la aplicación del trabajo de (Newell y Simon, 1972b) al diseño de interfaces. La idea principal tras este enfoque es que, al nivel en que es necesario estudiar la conducta en el caso de los interfaces, los mayores determinantes de ésta son los objetivos que el usuario tiene y la estructura que le impone la tarea para su descomposición y obtención de la solución. De ahí tenemos una visión de la solución de problemas basada en una jerarquía de objetivos y operadores que terminan por convertirse en acciones físicas que, efectivamente, avanzan en la dirección correcta. Ahora bien, debido a la necesidad de establecer los métodos a seguir, su proyecto debe limitarse al análisis de la ejecución de usuarios expertos que no cometen errores[2], ya que este proceso es el que se compone de secuencias más acordes con el esquema señalado anteriormente. Un análisis más detallado de esta perspectiva será realizado en un capítulo posterior, en el que se ofrecerá una visión del modelo GOMS, el cual es el producto principal de esta visión.

2 Obviamente, los sujetos expertos sí cometen errores. En sus estudios Card et al. eliminaban los tiempos que los sujetos utilizaban haciendo y recuperándose de sus errores.

2. La ciencia psicológica acumulativa es el objetivo, pero otro tipo de ciencia acumulativa

La perspectiva de Card et al (1983) tiene como limitación el dar lugar a un proyecto con objetivos relativamente estrechos, limitados a ciertas conductas y a ciertos usuarios. Aún sin rechazar que el éxito dentro de estos márgenes es suficientemente valioso por sí mismo, lo cierto es que la Psicología y otras ciencias pueden proporcionar información interesante a niveles distintos y también pueden ofrecer sugerencias acerca de aspectos que los trabajos de Card et al. dejaban sin considerar.

Para Lewis (1990) la cuestión estriba en que otros paradigmas pueden dar lugar a una ciencia acumulativa que no siguiera los esquemas de Card et. al. Este paradigma podría por ejemplo centrar sus esfuerzos en el estudio de los errores a partir de teoría psicológica relevante para este caso[3].

3. Lo importante es el contexto

Whiteside et. al. (Whiteside, et al., 1988; Whiteside y Wixon, 1987) expone la necesidad de tener en cuenta el contexto de uso de los artefactos diseñados para valorar de una manera realista su verdadera usabilidad. Para estos autores, las estrategias que se centran en estudios de laboratorio o en aplicación de teoría más o menos desarrollada son incapaces de captar muchas cuestiones que en la vida real pueden verdaderamente marcar la diferencia entre productos utilizables o no.

Un ejemplo que ilustra sus opiniones es el siguiente: En evaluaciones de hardware realizadas en el laboratorio, los usuarios encontraban el material ya dispuesto sobre un escritorio para empezar a trabajar con él. No obstante, realizando evaluaciones posteriores en el mismo lugar de trabajo descubrieron que en varias ocasiones el material era estropeado al desempaquetarlo debido a la forma en que éste se hallaba envuelto. Una modificación del envoltorio eliminó este tipo de accidentes.

Este tipo de cuestiones han llevado a considerar la utilización de métodos, que Monk (Monk, et al., 1993) denomina etnográficos, en la recogida

3 En este apartado, Lewis señala los trabajos de Norman como representativos. No obstante, esta apreciación no nos parece adecuada por lo que hemos preferido incluir algunos comentarios acerca de éstos en la sección dedicada a comentar que lo importante son los sistemas ya realizados como forma de proponer sugerencias para crear nuevos dispositivos.

de información para su uso en HCI. Algunas de las críticas realizadas al uso de los métodos tradicionales experimentales en este tipo de situaciones son:

1) el método experimental no tiene en cuenta el contexto real de la conducta observada,

2) el método experimental no pone la debida atención a la riqueza (o detalle) de la conducta humana, favoreciendo la observación detallada de un pequeño conjunto de categorías estrechamente definidas,

3) el método experimental no captura el significado subjetivo de la conducta siendo observada.

Por el contrario, los defensores de la utilización del método experimental se defienden al entender que existe una diferencia entre experimentación pura y experimentación aplicada, estando esta última más cerca de la utilización de variables de interés y en situaciones más realistas. Además, frente al método etnográfico puro, la generalización de los resultados es más posible que cuando toda la información es extraída a partir de interpretaciones subjetivas por parte del investigador y se presta atención al uso de análisis estadísticos y a la objetividad de las observaciones.

4. Lo importante es el proceso

Para los defensores de esta perspectiva, los esfuerzos teóricos a la hora de definir un diseño pueden siempre ser cuestionados en cuanto a su utilidad. Por el contrario, independientemente de lo bien planteadas que estén las ideas iniciales y el formato en que lo hayan sido (modelos formales por ejemplo) el peso de obtener un producto de calidad reside en la coherencia del proceso, las pruebas de campo y las modificaciones iterativas que eliminen las asperezas del interfaz hasta conseguir su usabilidad óptima.

Gould (Gould, 1987) defiende en primer lugar un análisis del usuario muy previo al comienzo del diseño. Ese focalizarse previo a la tarea de diseño debería tomar la forma de acudir a los lugares en que los usuarios realizan sus trabajos, estudiar sus necesidades con claridad, comprobar la aceptación de las nuevas ideas que se barajan, etc. Posteriormente, evaluaciones iterativas con el sistema deberían irse produciendo de modo que las mejoras pudieran ser incorporadas antes de tener productos demasiado acabados.

Un problema de gran interés enfrentado por los representantes de esta línea es cómo lograr coherencia dentro del proceso de diseño y además dejar constancia de los logros obtenidos mediante él. Dado que en muchas ocasiones, este proceso aparece como obvio a las personas que ven el producto final y no les resulta fácil apreciar el esfuerzo realizado y las ventajas sobre otras soluciones posibles, un registro de las opciones consideradas y las razones que respaldan su rechazo o aceptación pueden ser fundamentales para justificar los recursos empleados. En la actualidad existen notaciones semiformales que permiten asistir en este problema (v. p.e. el volumen 6 de Human-Computer Interaction dedicado especialmente al tema), las cuales permiten llevar registro de los objetivos planteados, los métodos para su evaluación en su caso y los logros obtenidos, todo lo cual permite demostrar el valor de las actividades realizadas durante el proceso llevado a cabo, así como guiar el proceso de una manera consistente sin apartarse de los objetivos establecidos.

5. Lo importante son los sistemas

Una perspectiva teórica probablemente muy extendida pero que no siempre es consciente para los que la mantienen es que la forma óptima de conseguir mejores dispositivos es simplemente observando los ya existentes y reflexionar acerca de ellos para proponer formas de mejorarlos.

Carroll et al. (Carroll y Rosson, 1991; Carroll, et al., 1992) han propuesto una estructura que pretende dar empuje a esa idea, de tal modo que se pueda construir una base de datos acerca de posibles ideas de diseño y su éxito o fracaso tanto en productos en fase de investigación como ya desarrollados. De este modo, los diseñadores podrían tener acceso a ideas manejadas anteriormente por otros diseñadores y aprovechar las experiencias que estos tuvieron con ellas.

Monk y Whright (Monk y Whright, 1991) proponen en la misma línea que los diseñadores registren pares "observación-invención" en el que se incluya: 1) Una generalización en relación con la conducta del usuario (observación), 2) Una invención (que indique implicaciones para el diseño y proporcione información heurística para posteriores invenciones y 3) Los datos que inspiraron la observación-invención (origen). Por ejemplo, un origen sería que al borrar una palabra en un texto, los usuarios tienen que eliminar la palabra y luego el espacio en blanco extra que queda después. El diseñador puede inventar el "borrado de palabra inteligente" que elimina, además de la

palabra, uno de los espacios extras. La generalización podría ser que al borrar objetos a menudo existen estructuras innecesarias relacionadas con ellos que también deberían ser borradas, por lo que el diseñador debería poner los medios para atender esta posibilidad.

Donald Norman ha llevado a cabo una línea de trabajo que se caracteriza por dos aportaciones fundamentalmente (Norman, 1981, 1983a, 1983b, 1984, 1986, 1988; Norman y Draper, 1986): un análisis racional de los errores cometidos por los usuarios y una teoría en estadios de la acción humana que permite la consideración de ayudas apropiadas a cada estadio.

Por ejemplo, Norman define la intención como la especificación al nivel superior de una acción deseada. Así, los errores en la intención (la acción deseada no es la correcta) los denomina "confusiones" (mistakes) y los errores en lograr realizar la acción se denominan "patinazos" (slips). A partir de ahí propone una clasificación de los "patinazos" que incluye las siguientes categorías:

- Errores de modo: Hacer la operación apropiada en un modo cuando de hecho estás en otro.

- Errores de descripción: La operación apropiada puede ser confundida muy fácilmente con otras disponibles.

- Errores de captura: Cuando una acción es similar a otra que se ejecuta muy a menudo o que está muy bien aprendida, aquella puede ser puesta en marcha y llevar a errores.

- Errores de activación: En este tipo de errores puede ocurrir que no se lleve a cabo la acción apropiada o que se realicen acciones inapropiadas en lugar de las correctas. Un ejemplo es el del olvido de la intención durante la ejecución de una acción debido a una interrupción u otra causa.

Norman propone una serie de ayudas y elementos que contribuyan a evitar este tipo de errores como son, por ejemplo, eliminar los modos en la medida de lo posible o aumentar al máximo su notoriedad cuando estén presentes (con cambios de color o elementos gráficos diferentes).

1.5. Una descripción del proceso de diseño que integre las diversas aportaciones y su relevancia en cada ocasión

Los enfoques teóricos anteriores pueden ser ligados a un esquema de las fases propuestas en relación al proceso de diseño de un interfaz (v.p.e. Brown, 1994) Este esquema, mostrado en la figura 1.2, ilustra como las diversas formas de generar teoría tendrían un impacto o aplicabilidad sobre una o varias fases. A continuación pasaremos a explicar los detalles de esta figura.

En el lado izquierdo se encuentra las fases en relación con el desarrollo de sistemas y con la especificación del interfaz. La primera parte (sombreada en gris) se encuentra solamente con el propósito de completar la figura y no será comentada. La segunda parte (especificación del interfaz) incluye las siguientes fases:

1) Análisis de requisitos y tareas: En esta fase, el diseñador se encuentra involucrado en la determinación del contexto de trabajo, las necesidades, los deseos y otro tipo de cuestiones de los usuarios finales. En ella, el diseñador debería analizar fundamentalmente qué funciones requieren los clientes y también la forma en que estas deben ser utilizadas para guiar las siguientes fases del diseño.

2) Especificación y evaluación de diseños previos: En esta fase, el diseñador empieza a plantear posibles interfaces hombre-computador que contemplen las necesidades de sus clientes. Usualmente, representaciones basadas en papel que utilicen notaciones de diseño de diálogos (v. capítulo 2) permitirán especificar el comportamiento del sistema interactivo. Además, dibujos de pantallas y otros métodos complementarán lo anterior. Una gran cantidad de evaluación previa debería ser realizada aquí, determinando posibles fallos, viabilidad, complejidad etc. del interfaz. Por otro lado, modelos del usuario permitirán obtener una apreciación previa del diseño, en fase de realización, que permita considerar, previamente al desarrollo de sistemas reales, la existencia de errores desde el punto de vista del usuario.

3) Desarrollo y evaluación de prototipos: A partir de cierto momento, los diseños realizados en papel y lápiz y analizados desde un punto de vista formal empezarán a convertirse en prototipos funcionando de manera limitada. La existencia de herramientas informáticas rápidas (p.e. Hypercard) o sistemas de programación dirigidos al interfaz (UIMS: User

Interface Management Systems) permitirían experimentar con sistemas sin necesidad de realizar toda la programación final. Progresivamente, el sistema final iría incorporando funcionalidad real hasta convertirse en un sistema completo.

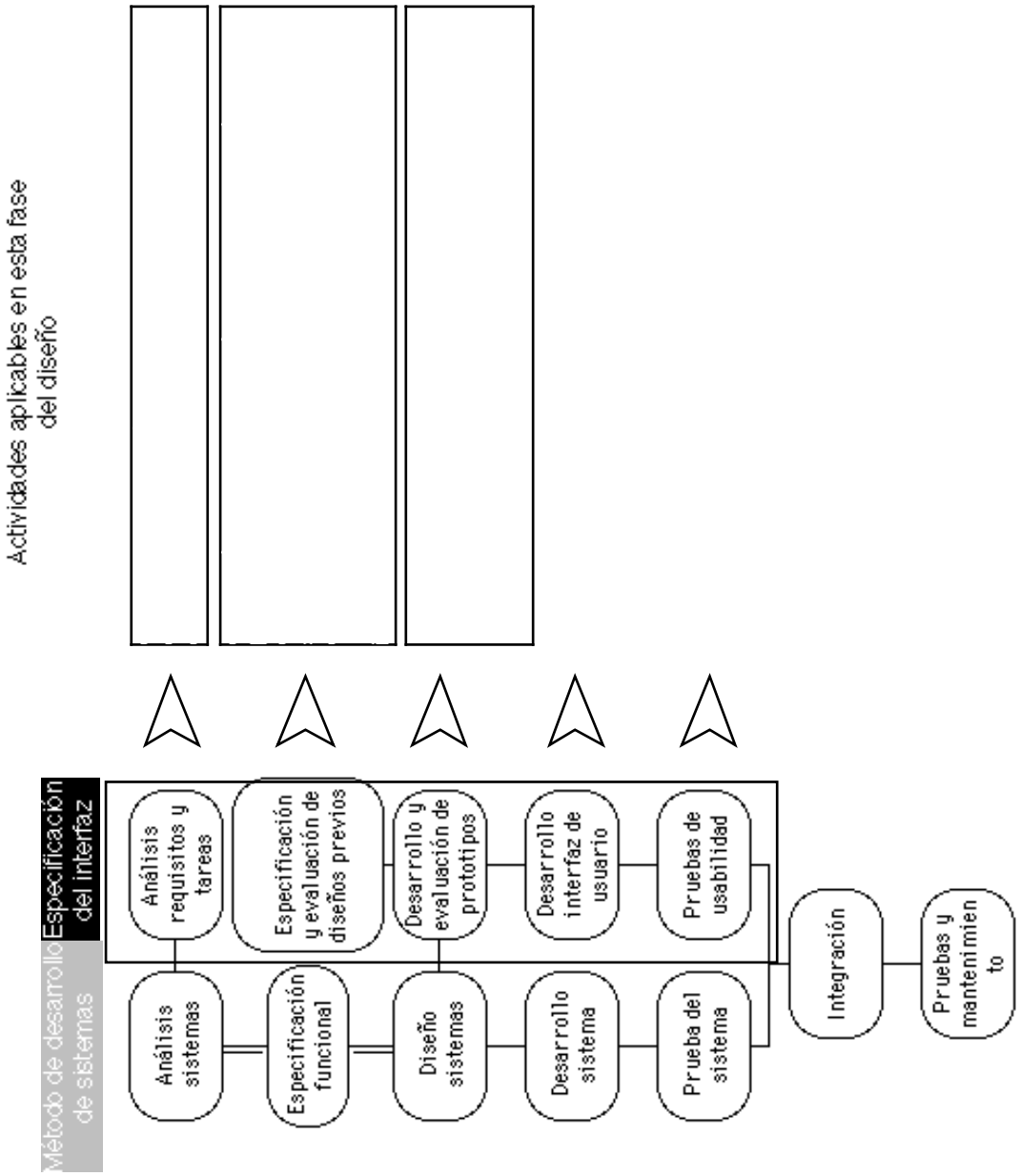
4) Desarrollo del interfaz de usuario: En esta fase las decisiones acerca de la forma del interfaz han sido tomadas y el problema aquí es lograr que el equipo de desarrolladores tenga una noción coherente de la forma que éste debería tener.

5) Pruebas de usabilidad: El sistema final debería ser evaluado empíricamente. Esta evaluación permitiría corregir detalles finales aunque es necesario destacar la poca relevancia que las sugerencias de diseño realizadas en esta fase suelen tener y la tendencia a desecharlas debido a la gran dificultad de corregir un producto ya terminado. También, se debería dejar constancia para el aprovechamiento en futuras ocasiones de los principios considerados en relación con el sistema para su posible reutilización en el futuro (p.e. ¿los gráficos animados sirven para llamar la atención?).

En la parte derecha se indica por medio de unas marcas la relevancia que tienen las distintas aportaciones teóricas en función del momento de diseño considerado. Tres marcas han sido consideradas:

RELEVANCIA... ☐Ninguna. ▒Parcial. ■Total.

Como es posible ver, los métodos formales son importantes fundamentalmente en una fase del diseño. Retomaremos esta cuestión en el siguiente capítulo que tratará acerca de una introducción general a los métodos formales.


Actividades aplicables en esta fase del diseño
Método de desarrollo de sistemas
Especificación del interfaz
Análisis sistemas
Especificación funcional
Diseño sistemas
Desarrollo sistema
Prueba del sistema
Análisis requisitos y tareas
Especificación y evaluación de diseños previos
Desarrollo y evaluación de prototipos
Desarrollo interfaz de usuario
Pruebas de usabilidad
Integración
Pruebas y mantenimiento

Capítulo 2
Los métodos formales en HCI

2.1. Introducción

Como veíamos en el capítulo anterior, los métodos formales eran técnicas que se aplicarían en el proceso de diseño de un interfaz hombre-computador antes de empezar la tarea de construir prototipos que implementen las ideas consideradas. De este modo, se podría esperar unos beneficios en términos de un producto con más calidad con una menor cantidad de esfuerzo y menor cantidad de "ensayo y error".

Formalizar un concepto puede ser concebido como la tarea de concretar, precisar o explicitar ese concepto. Esta formalización puede utilizar un mecanismo de representación del concepto que libere al resultado obtenido de la ambigüedad que generalmente se atribuye al lenguaje natural. Dentro de los métodos de representación, los provenientes de las matemáticas o la lógica han ocupado siempre una posición preponderante. En fechas recientes, el avance de las Ciencias de la Computación ha sugerido nuevos mecanismos de representación tales como los propios lenguajes de programación, reglas de producción para el almacenamiento del conocimiento, etc.

Los métodos formales en Ciencias de la Computación surgen según Dix (Dix, 1991), como una respuesta a la denominada crisis del software, un término relacionado con la aparición de grandes programas muy difíciles de depurar, mantener y modificar. Los métodos formales trajeron la promesa de especificaciones conceptualmente más claras que permitieran atajar el problema por medio de aproximaciones rigurosas.

Este tipo de beneficios también atrajo a los investigadores interesados en especificar interfaces hombre-computador y ya a principios de la década anterior Reisner (Reisner, 1981), inspirándose en conceptos lingüísticos intentó describir las reglas que guiarían la generación de sentencias correctas en un lenguaje interactivo.

No obstante, como señala Dix (1991), hay algo de extraño en la mezcla entre métodos formales, los cuales son percibidos como secos e insípidos y el diseño de interfaces, el cual tiene una apariencia mucho más alegre y colorida. A primera vista, parece dudoso que esta fusión pueda llevar a un resultado equilibrado, en el cual alguno de los dos lados de la balanza no empiece a pesar más que el otro, restándole su importancia. Probablemente, este riesgo es más real en cierto tipo de métodos formales (los más cercanos al lado psicológico y por tanto más dados a plantear premisas y supuestos menos robustos) que en otros (los más cercanos al sistema). Por ello, antes de seguir con la exposición acerca de los beneficios anticipados por los métodos formales es necesario exponer una clasificación de estos métodos que centre la discusión en cada uno de ellos.

Los beneficios esperados por el hecho de utilizar métodos formales serían:

1) Mayor rigor en la especificación.

2) Posibilidad de realizar comprobaciones y pruebas sobre las especificaciones.

3) Posibilidad de realizar modificaciones sobre las especificaciones con un coste más reducido que si el sistema hubiera sido construido y se tuviera que modificar posteriormente.

4) Facilitar el desarrollo, quizás teniendo herramientas que pasarían del modelo formal a una especificación ejecutable o algo similar.

Ahora bien, dentro de la etiqueta de métodos formales en HCI existen diversos modelos, los cuales pueden ser clasificados de modo amplio en tres tipos, que asumen de diferente manera este compromiso de rigor, comprobaciones, modificaciones y guía del diseño. Veremos a continuación una breve exposición de estos tipos de formalismos para luego retomar la cuestión de las ventajas que son capaces de asumir.

2.2. Una clasificación de los métodos formales en HCI.

Existen muchas maneras de clasificar los métodos formales en HCI (de Haan, et al., 1991; Farooq, 1988; Gugerty, 1993; Johnson, 1992; Kirakowski y Corbett, 1990; Monk, 1993; Simon, 1988). No obstante, pensamos que la realizada por Dix (1993) es la más completa ya que el resto tienden a centrarse en alguno de los tres apartados considerados por éste.

Dix (1993) distingue entre notaciones que intentan describir al usuario de un sistema informático, aquellas que describen el lenguaje de interacción utilizado por el usuario para transmitirle sus peticiones u órdenes y, por último, aquellas que reflejan los principios abstractos del sistema descrito. Estas categorías son comentadas más ampliamente a continuación:

a) Modelización del usuario.

Son sistemas que fundamentalmente intentan realizar descripciones centradas en el usuario antes que en el dispositivo. Dentro de esta área a su vez se distingue entre:

1. Descripciones Gramaticales: Son descripciones que intentan marcar las reglas o normas que los usuarios deben de conocer acerca de la sintaxis del lenguaje de diálogo.

2. Reglas de Producción: Describen la forma de alcanzar objetivos por medio del sistema a través de una serie de acciones. Mientras que las descripciones anteriores sólo permiten especificar las acciones permitidas para utilizar el interfaz, esta descripción da un paso más, al combinar esas acciones permitidas en grupos que realizan tareas de interés a los seres humanos.

3. Modelado cognitivo de los errores: Las arquitecturas cognitivas intentan realizar una descripción del uso de dispositivos por parte de seres humanos a partir de descripciones simplificadas de su funcionamiento cognitivo.

Estas notaciones son aquellas en las que un psicólogo estaría interesado en utilizar y le permitirían representar el conocimiento y/o las acciones cubiertas/encubiertas que el usuario necesita para manejar un sistema. En este grupo, las dos primeras categorías tienen mayor tradición y desarrollo, mientras que la tercera ha surgido en fechas recientes como un intento de superar algunas de las desventajas de las anteriores (aunque los modelos en ella incluidos todavía necesitan explicitar más sus reglas de funcionamiento).

Los modelos del usuario reivindican las siguientes ventajas (de Haan, et al., 1991):

1) Especificar un diseño con mucha precisión, sin ambiguedad.

2) Automatizar partes de la implementación de un interfaz de usuario.

3) Aplicar métodos analíticos que permitan establecer índices de facilidad de uso, aprendizaje y funcionalidad.

4) Pueden ser utilizados muy pronto en el diseño para predecir los aspectos de usabilidad.

5) Pueden ser aplicados a un precio más bajo que otras técnicas que requieren algo más que una especificación. Aunque las pruebas empíricas todavía serían necesarias, ciertas cuestiones podrían ser analizadas previamente y con menos esfuerzo que con pruebas empíricas.

Otra ventaja comentada por otros autores (p.e Gugerty, 1993) es:

6) Sugerir nuevas ideas de diseño al permitir realizar razonamientos desde el punto de vista del usuario al llevar a cabo las descripciones.

Ocasionalmente también se sugiere:

7) Permitir crear programas que implementen este tipo de modelos y automaticen parte de las tareas de análisis.

b) Sistemas de descripción de diálogos:

Los sistemas de descripción de diálogos permiten establecer las reglas de funcionamiento del interfaz, de tal modo que el diseñador puede especificar qué acciones estarán disponibles a los usuarios y qué consecuencias tendrán. Al comienzo de la sección dedicada a estos sistemas aparece una descripción de la subclasificación realizada.

Los sistemas de descripción de diálogos permiten la especificación de la conducta de un interfaz desde un punto de vista fundamentalmente sintáctico. Esto significa especificar qué acciones el sistema está dispuesto a aceptar como input y las respuestas que proporcionará como output, así como las reglas que el usuario debe seguir para construir correctamente las instrucciones que transmite al dispositivo.

Uno de los modelos originalmente formulados para justificar la utilización de notaciones de descripción de diálogos fue el de la conferencia de Seehiem (Pfaff, 1985) En esta conferencia se asumió un modelo lingüístico que distinguía entre los tres siguientes niveles de descripción en un programa de ordenador (Dix, et al., 1993):

• Nivel léxico: El nivel más básico. La forma de los iconos en la pantalla y las teclas presionadas.

• Nivel sintáctico: La estructura y orden de los inputs y los outputs. En lenguaje humano la gramática de construcción de sentencias.

• Nivel semántico: El significado de la conversación en cuanto a su efecto sobre las estructuras de datos internas del ordenador. En lenguaje humano, el significado atribuido a la conversación.

Un dispositivo puede ser descrito por separado en esos tres niveles: El nivel léxico (aspecto de las pantallas, tamaño de letras, etc.) sería el más centrado en los componentes de presentación. El nivel sintáctico, en el que las notaciones que vamos a describir se encuentran más orientados, trataría de los componentes dinámicos de los diálogos. El componente semántico trataría por último de lo que el sistema realmente hace y por tanto puede ser tratado de un modo separado.

Esta separación en tres niveles presenta como una de sus justificaciones argumentos similares a los aducidos para separar datos y forma de acceder a los

datos. Ello permite modificar el código de una de las partes sin modificar la otra, permitiendo realizar avances según ritmos propios.

En el caso del diseño del diálogo es posible incluso especificar la conducta (y con la herramienta apropiada ejecutarla como un prototipo) sin tener el código aún escrito.

Según Curry y Monk (Curry y Monk, 1990b) el modelo lingüístico propugnado en la conferencia de Seehiem (Pfaff, 1985) ha sido criticado debido a que resulta difícil especificar a través de él diálogos que pueden ser interrumpidos para atender a otros diálogos y posteriormente retomados para ser continuados, tal y como ocurre habitualmente en los interfaces de manipulación directa. En concreto, como alternativa se propone una descripción basada en "agentes" (Abowd, 1991). En ella, un agente es un grupo de componentes que responden a eventos. En esta aproximación sólo se especifica la estructura interna de los agentes y las interrelaciones con otros agentes, por lo que la descripción del diálogo es mucho más abierta. Otro problema que soluciona mejor este concepto del diálogo es la explosión combinatoria que se produce al intentar reflejar en una descripción de un diálogo todas los posibles estados en que el sistema podría estar, dando lugar a descripciones más sucintas y manejables.

Las ventajas anticipadas por la utilización de sistemas de descripción de diálogos son(adaptado de Curry y Monk, 1990a):

1) Permite describir los aspectos más importantes del interfaz sin necesidad de atender a los detalles, por lo que la tarea de comprender el diálogo resulta mucho más fácil.

2) Permitiría crear herramientas que transformaran las especificaciones en sistemas ejecutables.

3) Permite al diseñador explorar un número de opciones diferentes, al anticipar el resultado sin comprometerse a grandes cantidad es de tiempo y esfuerzo en desarrollar sistemas completos.

4) Permitiría comunicar a otros miembros del equipo de trabajo fácilmente y sin ambigüedades las especificaciones del diálogo.

5) Permitiría realizar análisis de ciertos principios o cuestiones que podrían dar lugar a errores en el interfaz (por ejemplo, si todos los estados pueden ser alcanzados o si hay consistencia en el lenguaje de acción).

Para Firth y Thomas (Firth y Thomas, 1991) los esquemas de representación de la interacción hombre-máquina pueden clasificarse en tres tipos: esquemas que se concentran sobre el análisis de tareas del usuario, sistemas de representación de la sintaxis del diálogo y modelos que engloban los otros dos aspectos y dan, por tanto, una visión más completa del problema. De entre estos últimos destacan CLG debido precisamente a la detallada visión que ofrece de las áreas que trata. A partir de esta apreciación ellos desarrollan una gramática más claramente especificada, introduciendo una gran cantidad de detalles, una más clara interconexión entre los conceptos y una enumeración de las opciones posibles en cada apartado.

c) Modelos del sistema:

La especificación formal es una herramienta de la ingeniería de software que utiliza descripciones abstractas matemáticas del sistema con objeto de exponer propiedades que contribuyan a su usabilidad percibida. Los dos aspectos claves de esta técnica es el ser una descripción matemática relativamente abstracta del espacio de problemas seguida por la formulación de principios precisos de usabilidad, tal como predictibilidad o robustez, que den forma al diseño buscado (Young y Abowd, 1994).

Para Dix et al (1993) los modelos formales del sistema se diferencian de las notaciones para la especificación de diálogos en que su objetivo es describir lo que las acciones del usuario le hacen al sistema. Estamos por tanto describiendo el significado de las acciones del usuario, lo cual nos lleva a entrar en el terreno de la semántica como algo diferenciado de la sintáctica del interfaz, lo cual sería descrito por las notaciones de descripción de diálogos.

Ahora bien, hay que tener en cuenta que la semántica a especificar es la correspondiente al interfaz, y no al dominio de la tarea implementado por el programa de ordenador o dispositivo a analizar. Aquí estamos interesados en analizar las propiedades del interfaz de usuario, y, por tanto, dejamos de lado aquellas funciones que corresponderían a los requisitos solicitados por los usuarios.

Algunas de las propiedades del interfaz junto con aspectos que podrían ser analizados en relación con ellas podrían ser:

• Deshacer (undo): ¿Cómo ha de comportarse una función de deshacer la última acción? ¿Qué resultado debe producir un deshacer de deshacer? ¿Cómo puede especificarse la noción de historia de las acciones? Un tema de bastante interés es como debería funcionar la función de undo en un programa de tipo colaborativo en el que varios usuarios interaccionan simultáneamente (Young y Abowd, 1994)

• Observabilidad: Muchos programas ahora afirman que son WYSIWYG (What You See Is What You Get), pero...¿qué significa realmente esto? Obviamente, en los últimos años hemos visto que lo que hay en la pantalla se parece bastante a lo que luego se imprime, pero un análisis formal de esta afirmación nos lleva a que estamos muy lejos de alcanzar este objetivo. Por ejemplo, programas que manejan grandes cantidades de datos (procesadores de textos) no pueden mostrar en la pantalla la información que manejan. Esto nos llevará, como se verá más adelante, a tener que plantear una serie de mecanismos (estrategias) que salven esta dificultad.

• Instantaneidad: Muy pocos programas tienen la posibilidad de responder de una manera instantánea aunque cada vez más aparecen muchos que parecen estar diseñados como si esto fuera posible. Quizás escribir cualquier tipo de texto sea lo más cercano a esta situación, ya que oprimir cada tecla hace aparecer una letra inmediatamente en pantalla. Programas de dibujo, de cálculo o de otro tipo no son tan afortunados. Sin embargo, actualmente existe un empuje muy fuerte en hacer esos programas interactivos, en los que una acción del usuario debe ser contestada por una acción del sistema sin posibilidad de realizar bloques[1] de acciones. Un análisis formal de estas situaciones nos haría reconsiderar qué forma y qué mecanismos se deberían contemplar en los programas

---

[1] Actualmente este fenómeno está alcanzando un dominio que, como usuario de este tipo de programas, afecta enormemente al responsable de estas páginas: los paquetes estadísticos. Tradicionalmente manejados por comandos actualmente se está realizando la transición a sistemas de manipulación directa, lo cual muchos usuarios cuestionan que sea verdaderamente beneficiosos (v.p.e. Wilkinson, 1989).

Los beneficios esperados por realizar descripciones por medio de estas notaciones serían (Harrison y Timbleby, 1990b):

• Obligan a una especificación muy precisa de los conceptos, lo cual permite una mayor claridad de ideas.

• Aclarar el verdadero significado de ciertas afirmaciones hechas en la industria, tal y como las comentadas anteriormente (WYSIWYG, deshacer, etc.).

• Corrección: Cuando un sistema falla lo puede hacer por dos razones. Bien por que no satisface los requisitos o bien por que falla algorítmicamente. El primero de los problemas es en muchas ocasiones debido al un problema de falta de formalización (formality gap) en los requisitos, siendo estos vagos y poco precisos. Los métodos formales afrontan estos problemas permitiendo: a) ligar partes de la especificación a partes concretas de código y b) permitiendo llevar a cabo pruebas mecánicamente mostrando que el programa coincide con la especificación.

• Tratabilidad: Una descripción formal permite probar si ciertas propiedades deseables en sistemas son realmente soportadas por la especificación realizada.

Una pregunta interesante para hacerse es la de quién podría estar interesado en la utilización de modelos formales de descripción del sistema. Harrison y Abowd sugieren que existen al menos dos casos posibles (Harrison y Abowd, 1991):

• Desarrolladores en pequeñas organizaciones poco interesados en documentar los sistemas que realizan y más interesados en utilizar los últimos avances que la industria puede ofrecerles y en experimentar sobre los sistemas que desarrollan.

• Programadores en organizaciones grandes que intentan documentar su trabajo, pero se encuentran limitados debido a que las descripciones en lenguaje natural suelen ser limitadas.

Los programadores del primer caso pueden estar interesados en la especificación de modelos formales por que esta tarea implica a menudo empezar a pensar acerca de los problemas de modo más riguroso y a la vez más novedoso por lo que puede ayudarles a plantear mejoras y enfoques que de otro modo quizás no serían capaces de ver.

Los programadores del segundo caso se enfrentan al problema de documentar y dejar especificadas las decisiones y opciones tomadas para que en el futuro esa información esté disponible en la organización.

Además, para llevar a cabo esa explicitación se utiliza algo más que el lenguaje natural, y se recurre a una notación o representación que permita de una manera sucinta llevar a cabo la especificación deseada.

Por último, la notación debería además de permitir describir el concepto, debería permitir extraer derivaciones de él, por medio de la utilización de una serie de operaciones admisibles que mostrarían implicaciones extraibles de los elementos básicos que no se encontraban directamente visibles a primera vista.

2.2.1. Como se relacionan los métodos formales en HCI con los modelos en Ingeniería y Ciencia

Los modelos en Ingeniería constituyen una forma de traer a la realidad conocimientos de tipo general para evaluar las consecuencias de la manipulación de ciertas variables sobre el resultado esperado. Es pues una manera de considerar la acción simultánea de muchos factores que en una aplicación de la vida real han de estar presentes y no se conoce con precisión las consecuencias de las posibles variaciones que en ellos pueden ocurrir. Por ejemplo, un modelo de un puente podría incluir factores acerca de materiales, resistencia, condiciones climáticas, etc. los cuales puestos en funcionamiento podrían dar una idea acerca del comportamiento del puente.

Los modelos son también utilizados con propósito científico para explicitar una serie de contenidos teóricos en afirmaciones más cercanas a la realidad, las cuales podrían ser más contrastables empíricamente y por tanto sujetos al rechazo o aceptación.

Los métodos formales en HCI son fundamentalmente modelos de tipo de Ingeniería. Los presupuestos científicos que sustentarían los modelos construidos por los métodos formales no son lo probado cuando estos métodos formales son utilizados. Aquellos presupuestos deberían ser probados independientemente, aprovechando para ello los recursos que el método científico a través del control experimental o otro tipo de metodología adecuada proporcionan. Lo que deben probar los métodos formales es su utilidad en las situaciones para las que han sido construidos. Es decir, si son capaces de representar el comportamiento de lo

modelado y anticipar consecuencias que resultan de la manipulación de sus componentes.

En ocasiones, los métodos formales en HCI son simplemente un mecanismo de representación o notación que permite visualizar de una manera aproximada un sistema en desarrollo. En los sistemas de representación de diálogos, por ejemplo, lo importante es lograr "contemplar" la forma en que finalmente el sistema funcionará, y no hay supuestos empíricos que necesiten ser probados independientemente.

2.2.2. Problemas para el uso de los métodos formales

Al utilizar métodos formales para representar un sistema interactivo concreto pueden darse un número de problemas que limitarían o impedirían la aplicación de éstos. Veremos a continuación algunos de ellos.

• Problemas de inadecuación de la notación elegida a los objetivos deseados. Uno de los problemas más obvios es utilizar una notación que no responda a las necesidades del proyecto. Por ejemplo, los modelos del usuario son interesantes para un diseñador interesado en analizar las consecuencias en ejecución y aprendizaje de los futuros usuarios de determinadas decisiones antes de incorporarlas definitivamente. Este tipo de análisis no podría hacerse con la mayoría de las descripciones del diálogo, ya que éstas están más dirigidas a mostrar el artefacto diseñado tal y como es sin tener en cuenta su uso por seres humanos. Por otro lado, ciertas pruebas que es posible realizar en este último tipo de métodos de descripción no podrían ser asumidas por los modelos del usuario (completud de la descripción, no existencia de acciones sin efecto, etc.). Los modelos del sistema, por otro lado, no servirían para describir sistemas concretos por lo que su uso estaría reservado para el avance de la teoría de los dispositivos.

La solución a este problema, aunque obvia tras una revisión adecuada de los métodos existentes, puede no serlo al principio de entrar en contacto con ellos. En nuestro caso, uno de los objetivos particulares de esta tesis se centró en la determinación de los métodos apropiados para alcanzar nuestro objetivo general, lo cual implicó una revisión exhaustiva de todos los métodos. Investigadores o diseñadores interesados en estas técnicas podrían consultar esta revisión a la hora de tomar decisiones similares.

• Problemas formales de descripción. Las notaciones utilizadas podrían ser analizadas de un modo formal con objeto de determinar su capacidad de descripción de todos los tipos de interfaces. Por ejemplo, dentro de los sistemas de descripción de diálogos existen notaciones que son poco adecuadas para representar sistemas con alto grado secuencialidad en la interacción tal y como ocurre en determinados sistemas de menús en los que todos los caminos están altamente predeterminados (p.e. un cajero automático) pero son adecuados para interfaces con poca secuencialidad y alto grado de libertad en la elección de la siguiente acción. Otros métodos en cambio se comportan de la manera complementaria, funcionando bien cuando aquellos funcionan mal y viceversa.

Otros ejemplos afectarían a los modelos de usuario. Por ejemplo, estos modelos suelen partir de un análisis que descompone las acciones de los usuarios en términos básicos hasta llegar a lo que es denominado "primitivas" o acciones más elementales. No existe en general una manera de determinar qué primitivas han de ser elegidas por lo que un elemento de ambiguedad es introducido siempre en este tipo de análisis. Otro ejemplo, es la capacidad de muchas notaciones de "sobreajustarse" a la conducta de los individuos, debido a las pocas constricciones acerca de la conducta de los individuos puestas en ellas. Este excesivo poder formal lleva a que prácticamente cualquier situación podría describirse utilizando mecanismos difícilmente justificables desde la Psicología.

Estos problemas podrían clasificarse en dos tipos dependiendo de su gravedad. Incapacidad para afrontar una situación a pesar de estar potencialmente incluida dentro de su rango de aplicabilidad, y, dificultad o ambiguedad a la hora de realizar la descripción debido a la falta de información adecuada. Este último problema será retomado en el siguiente punto y ahora seguiremos comentando la cuestión de la incapacidad de describir ciertas situaciones.

En este trabajo no hay mucho acerca de pruebas para demostrar la aplicabilidad de las técnicas desde un punto de vista formal. Dado que nuestro interés se centraba en realizar una aplicación similar a la que podría realizarse en una situación de diseño concreto, sin pretender que esta fuera peculiar en modo alguno, los problemas que era presumible encontrar eran más bien los de tipo práctico (no existe información clara acerca de como realizar las descripciones) antes que los formales. De hecho, y con esto realizamos una cierta anticipación, lo que hemos encontrado es que ciertos métodos ofrecen información suficiente para

realizar las descripciones, mientras que otros no, por lo que no ha sido posible realizar ningún tipo de descripción.

• Problemas prácticos de descripción. Para realizar las descripciones por parte de personas ajenas a los que propusieron los métodos a considerar, no basta con una exposición sucinta acerca de las propiedades y forma de realizar el análisis. Es necesario además información concreta que solucione situaciones especiales, ejemplos que puedan ser seguidos y, en general, una serie de soluciones quizás arbitrarias pero consensuadas que ayuden a tomar decisiones problemáticas. A veces la cuestión es todavía más simple, y los procedimientos propuestos son tan complicados que no son utilizados en la práctica salvo en ejemplos reducidos de demostración. Para comprobar este extremo la única forma de lograrlo es intentar repetir las descripciones realizadas por los investigadores.

• Problemas de predicción. Las metodologías consideradas realizan una serie de predicciones acerca de los interfaces hombre-ordenador. Estas predicciones son de tipo cuantitativo y de tipo cualitativo y permitirían tener una evaluación inicial acerca del sistema a construir de modo que, si estas se consideran negativas, puedan ser corregidas. Estas predicciones pueden , no obstante, no ser correctas o ser irrelevantes.

En realidad, los modelos del usuario han realizado aportaciones en el pasado demostrando lo acertado de las predicciones realizadas con sus modelos. No obstante, estas predicciones han sido realizadas dentro de contextos claramente experimentales en los que se manipuló la situación para hacer que los efectos esperados aparecieran de una manera saliente. Este tipo de efectos, como se ha visto en la revisión acerca de la interacción hombre-ordenador, encontrados en laboratorio no necesariamente aparecerán en situaciones reales por lo que, aunque la ciencia avance, la práctica aplicada puede que no se vea afectada. Es por ello de gran interés intentar mostrar que ciertos efectos que se encuentran de modo natural en el laboratorio aparecen igualmente fuera de él. Una metodología capaz de comportarse adecuadamente en el laboratorio no será utilizada fuera de él a menos que demuestre que es igualmente útil fuera de él.

Capítulo 3

Los modelos formales en HCI

3.1. Introducción

El siguiente capítulo presenta una revisión de una gran cantidad de notaciones y propuestas de métodos de descripción de los interfaces de usuario realizadas desde diferentes perspectivas.

La clasificación realizada sigue cercanamente a la propuesta por Dix et al (Dix, et al., 1993) y ha sido expuesta en el capítulo anterior cuando se trataron las cuestiones acerca de los métodos formales.

De todas las notaciones revisadas en este capítulo sólo tres han sido consideradas a la hora de intentar su aplicación tal y como se verá en el siguiente capítulo. No obstante, no queremos dar la impresión que las notaciones no consideradas carecen de interés. Al contrario, pensamos que muchas de las sugerencias presentes en la literatura tienen un grado de interés comparable entre sí, y situado al mismo nivel que las notaciones elegidas. Sin embargo, una notación, como muchas otras herramientas formales, necesita de algo más que simplemente su establecimiento en términos escuetos. Es necesario también que los autores y

otros investigadores proporcionen ejemplos, den soluciones a problemas avanzados y sean capaces de publicar explicaciones exhaustivas que otros puedan seguir. Estas condiciones no están, lamentablemente, lo suficientemente presentes tantas ocasiones como para que el número de notaciones realmente utilizables sea muy amplio. Además, incluso cuando estas condiciones se producen, el tiempo necesario para alcanzar un grado de destreza relativamente adecuado para su utilización es prohibitivo si se desea aprender a manejar muchas de las notaciones.

A pesar de este hecho, que podría quizás haber llevado a enfocar este capítulo como uno que simplemente describiera las notaciones utilizadas en el resto de este trabajo, hemos preferido, por contra, mantener el largo trabajo de recogida de aportaciones y presentar un capítulo que, somos conscientes de ello, resulta quizás en exceso ambicioso. Las razones que nos han llevado a esta decisión son las siguientes:

a) No somos conscientes en este momento de una aportación de este tipo en lengua castellana. De hecho, su extensión, rivaliza con las revisiones más amplias realizadas en lengua inglesa y, además, aportan cierta cantidad de trabajos de reciente publicación, sobre todo los provenientes de Amodeus (v. sección referida a Amodeus dentro de este capítulo). Por ello, consideramos que es un trabajo que puede resultar valioso en sí mismo.

b) Aunque quizás no es obvio a partir de la clasificación realizada (modelos de usuario, sistemas de descripción de diálogos y métodos formales del sistema) existe un continuo que iría desde descripciones formales de seres humanos-usuarios hasta descripciones de dispositivos mecánicos. Debido a la perspectiva desde la que este trabajo ha sido realizada, si nos hubiéramos centrado en las notaciones más cercanas a nuestra formación y que más tarde han sido analizadas en el siguientes capítulo, buena parte de las aportaciones podrían no haber sido incluidas. Esto no nos parecía totalmente satisfactorio. En nuestra opinión, una tesis sobre interacción hombre-ordenador debe realizar aportaciones que sean en cierto modo propias, centradas en este área e independientes de perspectivas particulares que la rodean pero no la definen completamente.

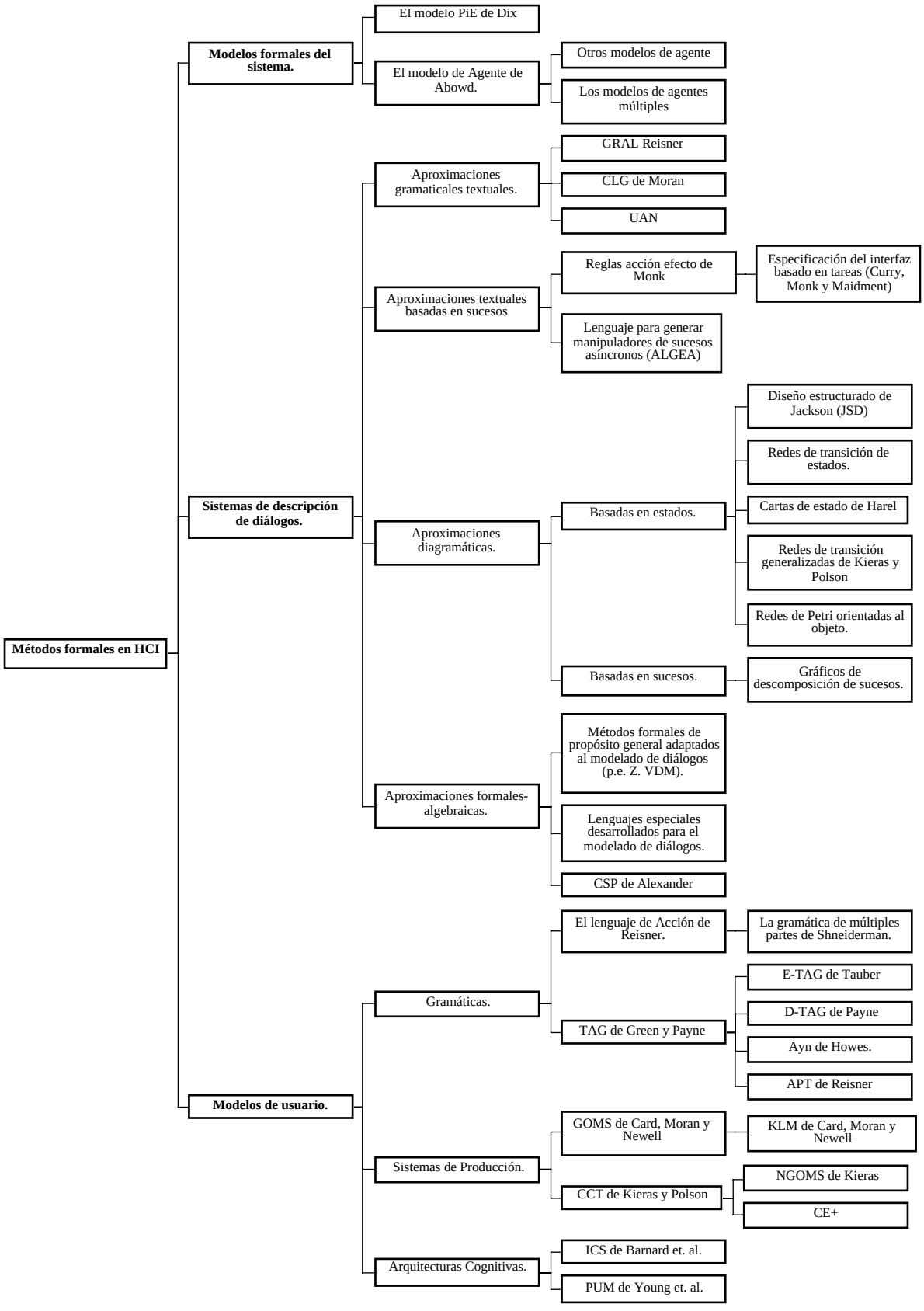


Figura 3.1: Diagrama de árbol de los distintos modelos considerados en este apartado.

Antes de entrar en la exposición de los distintos modelos parece conveniente advertir acerca de sus diferentes orígenes y tradiciones. De este modo, los sistemas de descripción de diálogos estarían más cercanos a una formación centrada en ciencias de la computación, mientras que los modelos de usuario serían más comprensibles desde una formación psicológica y/o en inteligencia artificial. Los modelos formales del sistema son también más cercanos a las ciencias de la computación aunque hay un grado de originalidad en su concepción y aplicación que les otorga un carácter también especial.

Para seguir con facilidad la discusión empírica del capítulo III es necesario adelantar que sólo tres modelos: TAG (Modelización del usuario: Gramáticas), NGOMS (Modelización del usuario: Reglas de producción) y *Action-Task* (Sistemas de descripción de diálogos: Especificación del interface basado en tareas) son los modelos considerados para realizar predicciones sobre un programa informático. Quizás alguien preferiría examinar con más detenimiento estos modelos y sólo revisar los aspectos más atrayentes de los otros modelos. De entre esos otros modelos, nuestra recomendación se dirigiría hacia GOMS, debido a su gran interés, tanto como origen de la corriente de trabajos en el área, como por la vigencia lograda a través de remodelaciones posteriores.

Un papel similar dentro del área de los modelos formales del sistema aplicados a los interfaces hombre-ordenador lo juega el modelo PiE de Dix.

Con objeto de facilitar la lectura, la figura 3.1 muestra una representación en árbol de los distintos modelos analizados así como sus orígenes en la jerarquía establecida. Esta clasificación, como se hará notar en el texto, no es sin embargo tan precisa como este gráfico sugiere, ya que algunas notaciones podrían haber sido situadas en lugares diferentes de los actuales.

3.2. Sistemas de descripción de diálogos

3.2.1. Introducción

Los sistemas de descripción de diálogos permiten especificar qué acciones el sistema está dispuesto a aceptar como input y las respuestas que proporcionará como output, así como las reglas que el usuario debe seguir para construir correctamente las instrucciones que transmite al dispositivo. Esta es una descripción fundamentalmente sintáctica, dirigida a describir la forma en la que es posible construir cadenas de input correctas.

El plan de este apartado es el siguiente: veremos en primer lugar algunos de los problemas generales asociados a las descripciones de los diálogos. Luego veremos una clasificación de los métodos de descripción y a continuación una descripción (breve) de algunos de los sistemas propuestos.

3.2.2. Problemas y cuestiones generales asociados a las descripciones de los diálogos

Aunque en el capítulo anterior ya se avanzaron algunas de las cuestiones más críticas en relación a las notaciones de descripción de diálogos, a continuación se dará una relación más extensa.

3.2.2.1. Facilidad de uso

El documento DREAM (Curry y Monk, 1990b) enfatiza de un modo muy especial las consideraciones relacionadas con la facilidad de uso de las propias notaciones. Ellos parten de que notaciones excesivamente complicadas tienen pocas oportunidades de ser utilizadas en la práctica así que señalan entre otros los siguientes puntos clave. Las notaciones deberían:

- Permitir fácilmente la concepción de diseños nuevos. Es decir, soportar las fases iniciales en las que se está esbozando un sistema y lo que se desea es describir las grandes lineas de éste sin pretender ser riguroso.
- Permitir una representación que ayude a la implementación del diseño, soportando fases más avanzadas de éste.

- Expresar diálogos con múltiples ramas (*threads*), en los que se den interrupciones y posteriores continuaciones.
- Facilidad de comprensión: las notaciones deberían ser fácilmente comprendidas por los diseñadores que las utilizarán.
- Facilidad de cambio: de modo que sea posible añadir modificaciones en las especificaciones de una manera relativamente sencilla.
- Facilidad de comunicación: permitiendo hacer llegar las descripciones hechas por un miembro del equipo de trabajo a los otros miembros.
- Facilidad de aprendizaje: las notaciones deberían poder aprenderse fácilmente.
- Aceptación dentro de un contexto más amplio: una notación interesante para los diseñadores podría llevarles a aceptar la tarea de descripción de diálogos como una práctica más seria y responsable y, por tanto, as una atención cuidadosa de todo lo relacionado con ella. Notaciones difíciles y de poco valor práctico minan la credibilidad que estos métodos tienen y por tanto limitan su uso.

3.2.2.2. Análisis de los diálogos

Una de las ventajas más importantes de llevar a cabo la descripción del diálogo de un sistema antes de su construcción radica en la posibilidad de llevar a cabo análisis que lo mejoren. Veremos a continuación algunas posibilidades.

Completud.

Este análisis permite comprobar que todas las acciones posibles dan lugar a resultados aceptables. A menudo, los programadores están interesados en determinar si existen acciones que no han considerado y que pueden bloquear el programa o producir un resultado indeseado. Es muy usual que estos programadores ofrezcan sus prototipos a posibles usuarios para determinar si existen casos de este tipo. Una notación de descripción de diálogos permitiría explorar más cuidadosamente este extremo para determinar la existencia de estos cabos sueltos.

En lineas generales, una buena regla es que los inputs que no tengan interés no deberían producir ninguna respuesta del sistema como resultado. No obstante, mensajes de alerta o ayuda podrían ser interesantes en determinadas ocasiones.

Determinismo.

Una acción por parte del usuario sólo debería poder llevar a un resultado. En ocasiones, en fases iniciales del diseño puede ocurrir que varios resultados se desprendan de una acción, llevando a resultados inesperados.

Consistencia.

Es de esperar que la misma acción siempre produzca el mismo resultado. Asimismo, tareas similares deberían dar lugar a resultados similares. Dado que resulta muy difícil automatizar esto en una herramienta de diálogo, el diseñador debería analizar esta cuestión manualmente. No obstante, y debido a que la consistencia es en muchas ocasiones una propiedad percibida subjetivamente antes que definible objetivamente, sólo utilizando buenas dosis de sentido común (Reisner, 1993) y quizás también herramientas apropiadas para el análisis a un nivel cognitivo (Payne y Green, 1986) más propias del tipo de los modelos del usuario es posible llegar a un diagnóstico apropiado.

Alcanzabilidad.

Un estado inalcanzable es aquel que no tiene un camino que permita llegar hasta él. Estos estados son errores de diseño que deben ser, obviamente, corregidos.

En un caso menos extremo, existen estados *difícilmente alcanzables*. Estos estados son aquellos que no pueden ser alcanzados de una manera sencilla ya que el camino hasta ellos es excesivamente largo o difícil. Por ejemplo, el procesador de textos utilizado para escribir este texto permite cambiar la primera letra de una palabra de minúscula a mayúscula (para formatear correctamente un nombre propio por ejemplo) seleccionando la palabra y utilizando un comando de menú, que produce un cuadro de diálogo en el cual se selecciona la opción deseada y apretando el botón de confirmación. Este comando es relativamente difícil de alcanzar sobre

todo si es comparado con el esfuerzo que supone eliminar la letra y ponerla en mayúsculas con el teclado, por lo que no es muy usual su utilización[1].

No obstante, intentar evitar la existencia de estados difícilmente alcanzables puede llevar a la acumulación de estados de primer orden, accesibles directamente, con la consiguiente sobrecarga del usuario. La solución aquí es organizar los estados en función de su importancia o frecuencia de uso, poniendo los más habitualmente alcanzados en primer plano y los menos en segundo.

Una situación contraria ocurre con los estados que suponen un cierto peligro potencial. En ellos, el diseñador debería esforzarse para hacerlos difícilmente alcanzables para de este modo evitar errores involuntarios.

Reversibilidad.

Esta propiedad hace referencia a la posibilidad de regresar a un estado anterior del diálogo de una manera sencilla. Estados que no permiten un paso sencillo a un estado anterior merecen un análisis cuidadoso. Hay estados que simplemente no son reversibles (borrar un fichero) y por lo tanto el diseñador debería tenerlo en cuenta, y otros que sólo podrían serlo de una manera muy complicada por lo que el usuario debería ser advertido de las consecuencias de su acción.

3.2.2.3. Propiedades léxicas y de presentación

Las descripciones del diálogo que serán descritas no tienen previsto, en lineas generales, un método para especificar las propiedades de presentación (aspecto de las pantallas, elementos gráficos) ni los léxicos (teclas que deben ser utilizadas en cada caso, dispositivos de input). Esta forma de hacer las cosas es a menudo considerada como positiva, ya que corresponde con una visión del diseño que parte de la descripción de la funcionalidad, posteriormente la especificación del diálogo y, finalmente, se llevan a cabo los aspectos de presentación y léxicos.

---

[1] Lo que no se usa se olvida, así que hasta en las pocas ocasiones en que su uso estaría justificado es igualmente difícil que se utilice. Por otro lado, hay que tener en cuenta que en la ortografía inglesa los nombres de los títulos se ponen en mayúsculas por lo que este comando tenga mayor interés cuando se está escribiendo en ese idioma.

No obstante, para Dix et. al (1993) esta forma de enfocar el diseño de los diálogos es en ocasiones poco adecuada, ya que muchos aspectos relacionados con estos dos últimos puntos pueden determinar estructuras de diálogo distintas. Utilizar un ratón puede llevar a elementos de diálogo distintos que si se utiliza únicamente el teclado como elemento de input. Del mismo modo, una pantalla más pequeña puede limitar al diseñador y obligarle a interacciones distintas que las que pueden llevarse a cabo con ventanas más grandes. Al menos un sistema comercial cuyo punto fuerte radica en su interés como UIMS (User Interface Management System), destaca este mismo punto (Serius, 1992).

3.2.2.4. Estructuras de diálogos dinámicas/estáticas

En una estructura de diálogos estática al usuario no se le permite generar objetos de diálogo nuevos, tal y como nuevas pantallas o ventanas. En estos sistemas, el diseñador puede enumerar a la hora de realizar la especificación todos los posibles estados o lugares que el usuario visitará. En una estructura dinámica, el usuario puede crear un número indefinido de nuevas estructuras, algunas de ellas copia de otras ya existentes o incluso dotadas de características nuevas. Este tipo de diálogos suponen un reto para las notaciones de descripción.

Según Dix et. al (Dix, et al., 1993) muchas de las descripciones de diálogos sólo son capaces de describir estructuras estáticas, en las cuales, las posibles pantallas-ventanas-etc están predefinidas y el usuario no puede producir presentaciones radicalmente nuevas. Este problema para los autores no es especialmente importante, ya que muy pocos sistemas no tienen cada una de las posibles ventanas enumeradas previamente.

3.2.3. Clasificación de los sistemas de descripción de diálogos

Curry et. al. (Curry y Monk, 1990b) proponen una clasificación de los sistemas de descripción de diálogos basada en dos dimensiones: Características superficiales de las notaciones y orientación del modelado. Se consideran tres tipos de características superficiales:

a) Notaciones basadas en texto.
b) Notaciones basadas en diagramas.
c) Notaciones algebraico-matemáticas.

Mientras que la primera y segunda dimensión se distinguen por el hecho de utilizar notaciones textuales o basadas en gráficos, flechas, etc., la tercera se

diferencia de la primera en las bases más matemáticas del formalismo, las cuales permitirían pruebas de propiedades que no serían posibles con aquellas.

La dimensión de orientación del modelado distingue entre:

a) Aproximaciones basadas en estados.

b) Aproximaciones basadas en sucesos.

Las aproximaciones basadas en estados describen el sistema en el lapso de la ocurrencia de un nuevo suceso que modifica ese estado. Las aproximaciones basadas en sucesos en cambio son la otra cara de la moneda, ya que describen los sucesos posibles en un momento dado y los estados que esas acciones desencadenarán. La información expresada en términos de un estilo de notación podría ser transformada al otro estilo de notación bastante fácilmente.

Otras clasificaciones pueden ser encontradas en Abowd (1991), Dix, et al. (1993), Farooq (1988), Williges (1987) aunque son probablemente menos exhaustivas.

3.2.4. Descripciones de sistemas específicos

3.2.4.1. Aproximaciones gramaticales textuales

Los métodos aquí discutidos puede decirse que comparten el inconveniente de estar basados en un modelo conversacional de la interacción hombre-ordenador, lo cual los hace más apropiados para interfaces de lenguajes de comandos que para los de manipulación directa, mucho más usuales hoy en día.

3.2.4.1.1. Command Language Grammar (CLG) de Moran.

Este sistema podría ser clasificado en el apartado anterior de métodos de modelización del usuario. De hecho, su alcance es tan amplio que en realidad puede decirse que abarca todos los aspectos de la HCI. No obstante, su amplitud es probablemente también su mayor inconveniente.

Moran (Moran, 1981b) distingue entre 6 niveles de la interacción hombre-ordenador, sumarizados en la figura 3.2.

Componente conceptual	Nivel de tarea Nivel semántico
Componente de comunicación	Nivel sintáctico Nivel de interacción
Componente físico	Nivel de disposición espacial Nivel de dispositivo

Figura 3.2: Niveles de la interacción hombre-ordenador según Moran

Moran sólo analiza los dos primeros componentes, dejando el último para una elaboración futura o en manos de la "Ergonomía clásica" (de Haan, et al., 1991). Esta división en seis niveles corresponde a las fases por las que, según Moran, el usuario tiene que pasar para llevar a cabo una tarea. En concreto,

1) El nivel de tarea describe la estructura de las tareas que el usuario puede llevar a cabo con un ordenador. Esta descripción se realiza en CLG de un modo básicamente informal.

2) El nivel semántico es la descripción de los objetos que el usuario y el ordenador comparten y que permitirían llevar a cabo la tarea. Estos objetos tienen una apariencia distinta según el punto de vista del ordenador (estructuras de datos y "procedures") o del usuario (entidades conceptuales y operaciones sobre ellas).

3) El nivel sintáctico describe las entidades conceptuales y las operaciones que pueden ser realizadas en un contexto dado, la estructura de los comandos y su forma específica.

4) El nivel de interacción, lleva a cabo esa descripción a un nivel más detallado, especificando las secuencias físicas y la estructura de la interacción.

Con objeto de dar una imagen aproximada de esta notación se muestra la descripción a los tres primeros niveles (tarea, semántico y sintáctico) de la operación de examinar un mensaje en un sistema de correo electrónico.

```
NIVEL DE TAREA

NUEVO_CORREO=(TAREA (*Comprobar si hay mensajes  nuevos y leerlos. esta es
     la tarea más usual))

HACER (SECUENCIA: (COMPROBAR_NUEVO_CORREO)

     (LEER_NUEVO_CORREO))

NIVEL SEMANTICO (*Aquí se ha obviado la descripción de entidades
     y objetos conceptuales)

METODO_SEMANTICO=(*UN METODO SEMANTICO PARA LEER CORREO NUEVO)

HACER (REPETIR

        PONER_M_A (CADA MENSAJE CON EDAD=NUEVO)

        HACER (SECUENCIA: (MOSTRAR M)

                          (LEER M)

                          (OPCION BORRAR M)))

METODO_SINTACTICO=(*UN METODO SINTACTICO PARA LEER CORREO NUEVO)

HACER (REPETIR HASTA FIN_BUZON

      HACER (SECUENCIA: (LEER (MENSAJE_ACTUAL) EN AREA_MENSAJES)

      ELEGIR: (MOSTRAR_MENSAJE_SIGUIENTE)

              (BORRAR_MENSAJE_ACTUAL))
```

Figura 3.3: Un ejemplo de CLG de Moran (1981)

Los dos últimos niveles harían referencia los componentes más básicos, tal y como la organización de la pantalla, los colores utilizados, etc. y la forma de los dispositivos de input-output. Este nivel como hemos dicho no es analizado por Moran.

En el artículo original de exposición de su metodología se introduce el ejemplo de un sistema de correo electrónico que es descrito atendiendo a los cuatro primeros niveles.

Según Moran (1981), su sistema puede ser visto desde tres puntos de vista: 1) Psicológico: Según él, CLG es una descripción del conocimiento del usuario ideal. 2) Lingüístico: CLG es una descripción de la estructura de todos los lenguajes de comandos, y es por tanto adecuado para describir cualquier ejemplo. 3) Del diseño: Este método permitiría al diseñador llevar a cabo su representación usando una estrategia top-down, permitiéndole generar y evaluar alternativas de diseño previamente a su construcción.

Ha sido Sharrat (Sharrat, 1987a; Sharrat, 1987b) el autor más interesado en llevar a cabo evaluaciones de CLG. En concreto, él hizo que sus estudiantes lo utilizaran en una situación de diseño práctico, y encontró que su estructura era excesivamente rígida, de modo que llevar a cabo cambios se convertía en una tarea muy difícil.

Belloti (Belloti, 1988) entrevistó a diseñadores trabajando en situaciones reales y llegó a la conclusión que CLG era excesivamente trabajoso para ser utilizado en un contexto industrial desde el punto de vista de éstos.

Ademas, según (Curry y Monk, 1990b) las especificaciones no son fáciles de entender y comunicar entre diseñadores, así que el creador original se convierte en responsable de llevar a cabo todos los cambios propuestos. Además, sólo ha sido utilizado en proyectos experimentales, no permite especificar diálogos con varias ramas funcionando simultáneamente por lo que no es apropiado para interfaces de manipulación directa, asume conducta sin errores y no hay previsión de estos, los diseñadores sufren una gran cantidad de errores de especificación debido a su complejidad, su estructura es excesivamente rígida y las decisiones tomadas en niveles superiores influyen demasiado en los tomadas a niveles inferiores.

3.2.4.1.2. User Action Notation (UAN) de Hix y Hartson

UAN proporciona al diseñador una notación que le permite especificar las acciones de los usuarios, el feedback que el sistema les proporciona como respuesta y los cambios en el estado del sistema que se producen a continuación. Estos tres elementos se representan en una tabla en la que es posible ir viendo todos esos cambios paso a paso.

En la tabla 3.1 se muestra un ejemplo el cual hace referencia a seleccionar una opción en un menú tipo pull-down (al apretar desciende).

Tarea: seleccionar opción en menú pull-down. **Devuelve:** opción		
ACCIONES DEL USUARIO	FEED-BACK	ESTADO DEL SISTEMA
(~[X, OPCION]	X,OPCION!	
[X, OPCION]~)*;	X,OPCION-!	

~[X, OPCION´]	X,OPCION´!	
M^	X,OPCION´!!	DEVUELVE OPCION´

Tabla 3.1: Descripción de un menú pull-down en UAN.

El significado de los símbolos puede consultarse en la tabla 3.2:

Símbolo	Significado
[X,Y]	una posición en la pantalla
()	grupo de acciones
*	cero o mas ocurrencias
~[X, OPCION]*	mover el cursor un numero arbitrario de posiciones en la pantalla
!	seleccionar
-!	deseleccionar
!!	selección alternativa
Mv	apretar botón ratón
M^	soltar botón ratón
´	segundo estado
;	el usuario puede interrumpir la tarea en este punto.

Tabla 3.2: Significado de algunos símbolos utilizados en UAN

En el ejemplo anterior tenemos lo siguiente:

Línea 1: Se mueve el ratón sobre las opciones del menú sobre el que se van a elegir una opción. El feedback del sistema es que a medida que el ratón pasa sobre las opciones posibles el ítem correspondiente es iluminado.

Línea 2: Cuando el cursor del ratón se sale del objeto que es posible seleccionar el feedback del sistema es que el objeto se desilumina.

Línea 3: Indica que es posible para el usuario mover el cursor otra vez sobre el sitio anterior de modo que el objeto vuelva a estar iluminado.

Línea 4: Al soltar el botón del ratón el feedback del sistema es que el objeto que estaba iluminado se ilumina ahora utilizando una iluminación especial. Este estado actúa como un clave para el software de aplicación para informar que el menú ha sido seleccionado.

Para Curry (Curry y Monk, 1990b) esta notación presenta las siguientes ventajas.

- Es apropiado para interfaces de manipulación directa.

- Se realizan las descripciones de abajo arriba y se concentra sobre todo en el nivel bajo de la interacción. Acciones a un nivel superior pueden ser obtenidas agregando estas acciones más simples. No obstante, para estos autores, no está claramente especificado cómo hacer esto último.

- Su punto más fuerte se encuentra en la interconexión que ofrece entre el análisis de tareas y las notaciones orientadas a la implementación.

En cuanto a las desventajas citan las siguientes:

- Está muy orientada al estilo Macintosh ya que no permite especificar cambios de modo (lo cual se adapta al Macintosh).

- El diseñador no puede conseguir una visión general del diseño tal y como se podría obtener con otras técnicas.

- No permite considerar diseño de pantallas más que como figuras adicionales.

3.2.4.1.3. Grammatical Representation of Action Language (GRAL) de Reisner.

Este método, será revisado con más detalle en la sección dedicada a modelización del usuario así que sólo será brevemente comentada. Esta notación puede ser considerada un sistema de descripción de diálogos ya que permite una especificación efectiva y completa de éstos. No obstante, por otro lado, su autora

afirma que es posible extraer consecuencias acerca del comportamiento del usuario (Reisner, 1981) y por tanto puede ser considerado una herramienta de modelado del usuario.

3.2.4.2. Aproximaciones textuales basadas en sucesos

La idea de las aproximaciones basadas en sucesos se apoya en que el diálogo hombre-ordenador puede ser concebido como una serie de sucesos-acciones que disparan efectos, las cuales a su vez modifican los sucesos-acciones posibles.

3.2.4.2.1. Action-effect rules (AER) de Monk.

En Dix (1991) se describen las reglas de acción efecto como teniendo tres componentes:

• Una descripción de las posibles acciones disponibles al usuario.

• Una descripción textual o gráfica de las pantallas típicas disponibles.

• Para cada modo visual, un conjunto de reglas que describen como el sistema responde a cada posible acción del usuario.

Monk (Monk, 1990) comenta que la parte de efecto incluye el efecto observable sobre los objetos subyacentes siendo manipulables. Además, existen condiciones que muestran el contexto dentro del cual las reglas se aplican.

Para llevar a cabo una descripción en términos de AER el primer paso es especificar qué acciones podría hacer el usuario que tengan significado para el sistema. Las reglas de acción efecto son generadas listando los efectos que cada una de estas acciones tienen y el contexto o condición que controlará el efecto que puede ocurrir. El ejemplo que se muestra más tarde acerca de DSN (Dialogue Specification Notation) es válido también para la notación de *action-effect* debido a la similaridad entre ambas notaciones.

Estas reglas se prestan a una serie de análisis que permiten realizar algunas evaluaciones del sistema siendo diseñado. Dix (Dix, 1991) señala completud, predictibilidad del sistema, consistencia (o falta de ella), alcanzabilidad y reversibilidad (una forma de deshacer). Estos análisis son realizados por el diseñador utilizando su propia intuición.

Entre sus desventajas Curry et. al. (Curry y Monk, 1990b) señalan que:

• Es pobre a la hora de proporcionar una visión general de como las cosas encajan entre sí.

• No tiene componente de presentación. Aunque según Dix es posible realizar descripciones de la presentación de la pantalla, éstas tienen que ser realizadas de modo informal.

Las reglas *Action-Task* fueron el primer paso en un método denominado *especificación del interfaz basado en tareas* (Curry, Monk y Maidment, en prensa). el cual incluye otros componentes y una ligera reformulación de las reglas acción-tarea.

Esta notación intenta abarcar todo el proceso de la interacción hombre-ordenador, teniendo en cuenta no sólo cómo modelar la traducción de objetivos del usuario en acciones (es decir, los procesos cognitivos de los usuarios) sino también el como estas acciones repercuten en efectos del sistema (v. fig. 3.4).

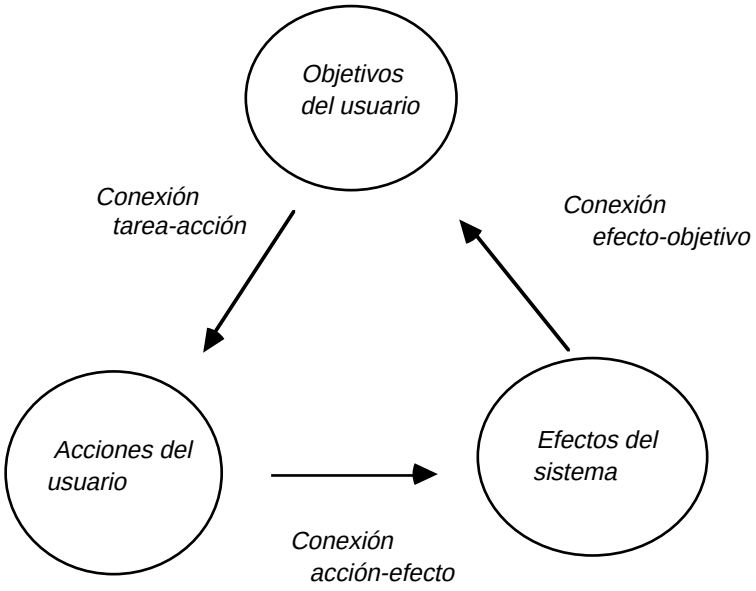


Figura. 3.4: Modelo de la interacción hombre-ordenador

Así, su modelo pasa por tres fases:

1) Modelado del contexto-tarea. El cual consiste fundamentalmente en una análisis de tareas desde el punto de vista del usuario.

2) Modelado de los diálogos. El cual consiste en un modelado del sistema.

3) Especificación de los detalles finales. En el último paso el diseño es completado y se proporcionan los detalles de lo que el sistema necesitará hacer para responder de la manera esperada. Además se comprueba que la descripción se ajuste a una serie de condiciones formales que garantizan su corrección.

1. Descripción del modelado del contexto-tarea.

Para realizar esta descripción se deberían realizar tres actividades: Una descomposición objetiva del trabajo a realizar (una forma de análisis de tareas), crear una lista de excepciones y un esquema de trabajos representativos.

a) *La descomposición objetiva del trabajo* es una forma de análisis jerárquico de tareas. En ella se hace una descripción cada vez más detallada de los estados que cada tarea particular debe intentar alcanzar. Este detalle intenta evitar que se de un excesivo énfasis a las prácticas de trabajo existentes si se hiciera una descripción en términos de procesos. También, la descripción intenta evitar incluir un plan que especifique el orden en que deben llevarse a cabo los objetivos y subobjetivos.

b) *Una lista de excepciones* indica posibles interrupciones en el flujo de trabajo identificado mediante la descomposición objetiva del trabajo. Esta descomposición es una descripción idealizada en el sentido de que no prevé la aparición de algún tipo de problema o interrupción en la conducta durante la ejecución de la tarea. En general, los diseñadores tienen en cuenta cuando pueden ocurrir interrupciones por parte del sistema pero no cuando estas provienen por parte del usuario. El sentido de elaborar una lista de excepciones explícita para los dos casos se encuentra en permitir a los usuarios una cierta flexibilidad en sus tareas cuando éstas ocurran.

c) Realizar *esquemas de ejemplos representativos* permiten dar una idea de como se producen las interacciones en la realidad, de tal modo que las descripciones formales realizadas posteriormente puedan ser probadas con ellos.

2. La notación de especificación de diálogos.

Dialogue Specification Notation (DSN) es una notación que modifica la del método de *action-effect*. Es por tanto una notación basada en reglas de producción la cual permite describir la forma en que el software de aplicación responderá a determinados sucesos. Estos sucesos pueden provenir bien del usuario o bien de la

aplicación y ocurrir en un contexto particular que también necesita ser especificado.

A continuación se puede ver un ejemplo de regla de diálogo realizado mediante DSN en el que un usuario comienza a utilizar una fotocopiadora. En la regla de diálogo hay una serie de precondiciones las cuales determinan si la regla es aplicable o no. En este caso las precondiciones que habría que comprobar es si se ha introducido una clave de usuario correcta[2] y si la fotocopiadora no está ya copiando. Si esto es así, el usuario puede enviar la orden de **comenzarcopiar** la cual cambia el estado del sistema de Nocopiando a copiando. En la parte inferior se muestra un ejemplo de posibles sucesos que tanto el usuario como el sistema podrían llevar a cabo así como los diferentes estados en los que el sistema puede encontrarse.

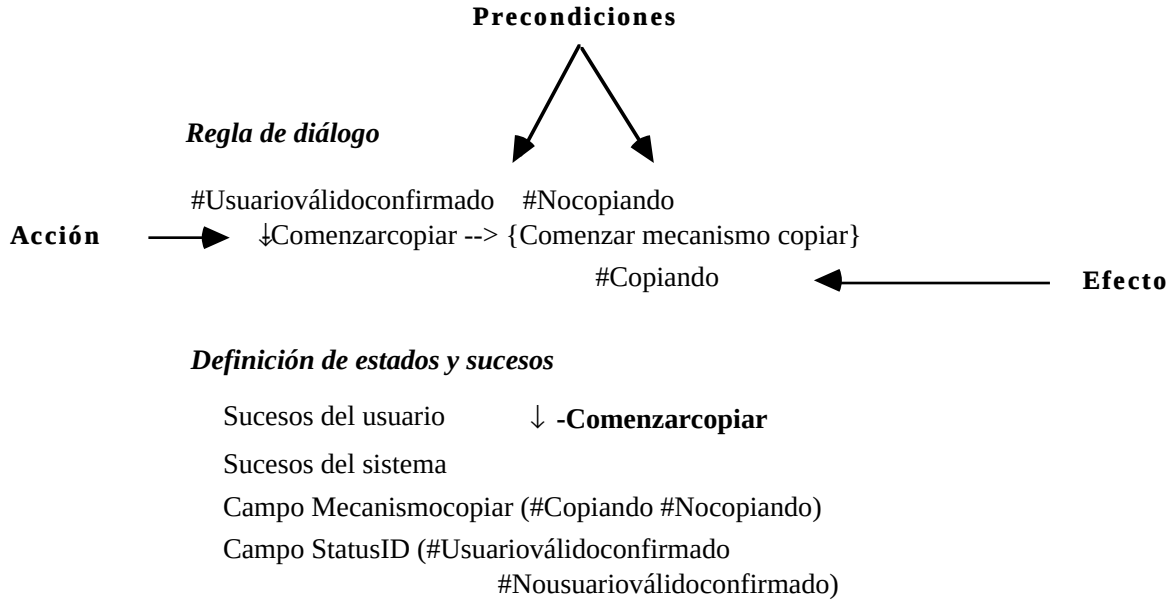


Figura 3.5: Un ejemplo de DSN

[2] Puede que la idea de una fotocopiadora que necesita una clave para ser utilizada resulte extraña pero en la visita realizada al departamento de Psicología de la Universidad de York, institución en la que el dr. A. Monk realiza sus actividades, el doctorando tuvo la oportunidad de utilizar esa fotocopiadora y, efectivamente, tuvo que utilizar esta clave.

3. Especificación de los detalles finales.

Las dos fases anteriores según los autores funcionan mejor si se producen de un modo relativamente no constreñido, buscando la originalidad y las ideas novedosas antes que la precisión. En esta tercera fase ya sería cuando las ideas anteriores tomarían un aspecto más concreto y además se verificaría que no faltan detalles.

Esta verificación significa comprobar que la descripción es internamente consistente y reúne ciertos requerimientos. Ello se hace por medio de una herramienta informática que chequea las descripciones, permite editarlas y además hacerlas funcionar. Las comprobaciones que lleva a cabo no obstante son muy generales, del tipo "condiciones duplicadas" o "violaciones de exclusividad mutua", etc.

Comentarios finales.

Este método tiene asociado un método de análisis de tareas, lo cual no es común en otros métodos.

Algunos inconvenientes que pueden ser señalados son los siguientes:

• Como método basado en reglas de producción, no resulta muy adecuado para describir interfaces de tipo secuencial ya que es necesario ir especificando una serie de reglas que, de una manera bastante monótona, van generando un efecto que, a su vez, permitiría generar el siguiente efecto. Una notación diagramática (tal como STN) es mucho más interesante en este caso. Por otro lado, su potencia está en un sitio en el que las notaciones diagramáticas no son adecuadas, ya que son capaces de representar la concurrencia de una manera relativamente simple.

• En el sistema informático proporcionado para realizar las descripciones, no es posible especificar reglas que dependan simultaneamente de más de una condición. Es decir, condiciones del tipo A o B no son posibles. Ello lleva a la necesidad de realizar especificaciones excesivamente complejas y difíciles de seguir.

• La noción de jerarquía en la descripción no está tratada explícitamente. Aunque de una manera natural, al realizar las descripciones, el diseñador tiende a introducir componentes de alto nivel sin entrar a especificar los detalles internos,

resulta difícil determinar qué capas resultan apropiadas en las situaciones específicas.

3.2.4.2.2. Language for generating asynchronous event handlers (ALGEA).

Según Curry et al. (Curry y Monk, 1990b) ALGEA es un lenguaje basado en el C, pero que contiene una serie de aditamentos dirigidas al procesamiento de sucesos. De esta notación, destaca el hecho de estar basado en el concepto de "manipuladores de sucesos" (event handlers) los cuales han sido utilizados en el modelado de los diálogos de un rango amplio de sistemas. Cada manipulador de sucesos es un proceso que permanece "dormido" esperando que se produzca el suceso para el cual ha sido preprogramado para responder. Este método presenta evidentes conexiones con la perspectiva de la programación orientada al objeto.

Las ventajas de esta perspectiva para Curry et. al. (Curry y Monk, 1990b) son las siguientes:

• Permite manipular interrupciones en el diálogo y diálogos de múltiples ramas (concurrencia).

• En general son muy similares a los sistemas de programación orientados al objeto, por lo que los programadores encuentran fácil comprender la lógica subyacente.

En cuanto a las desventajas señalan:

• Es difícil observar el lenguaje de input/output.

• Como los manipuladores pueden estar funcionando simultaneamente es difícil para los diseñadores saber qué es lo que está pasando en cada momento.

• Se parecen mucho al código de programación lo cual dificulta el aplicar análisis desde un punto de vista de diálogo.

Otra notación que adopta una forma semejante a la de un lenguaje de programación es Event Response Language (ERL) (Curry y Monk, 1990b).

3.2.4.3. Aproximaciones diagramáticas

Las aproximaciones diagramáticas se caracterizan por utilizar notaciones basadas en representaciones gráficas. Se distingue entre las notaciones basadas en estados y las basadas en sucesos.

3.2.4.3.1. Basadas en estados.

3.2.4.3.1.1. Jackson Structured Design (JSD).

JSD es una técnica que ha sido utilizada para en la programación y en el diseño de sistemas. Una parte de esta técnica, los diagramas JSD, han sido usados para el análisis de tareas y el diseño de diálogos. Estos diagramas permiten una descomposición jerárquica de un diálogo, sirviendo además para especificar si una determinada parte del diálogo es opcional o no. El número de diálogos que es posible describir mediante este método es de tamaño limitado pero incluye muchos sistemas basados en menús.

Un ejemplo que muestra la forma en que se podría representar el diálogo de conexión a una base de datos para realizar una modificación o cualquier otra tarea es mostrado en la figura 3.6. El diagrama tiene que ser leído de izquierda a derecha y tendría que ser autoexplicativo salvo por los símbolos situados en la partes superior derecha de ciertos recuadros. Estos son:

- * Indica iteración. Cualquier número de repeticiones.

- o. Indica opcionalidad. Las tareas marcadas con una "o" pueden o no ser realizadas.

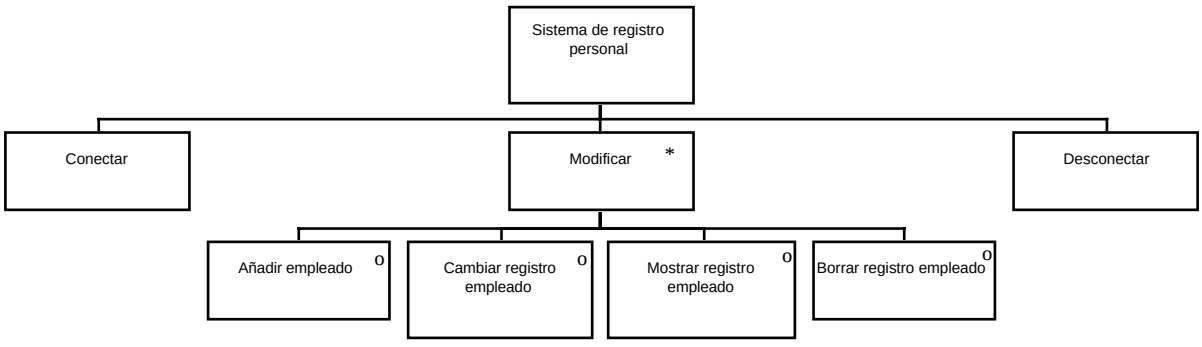


Figura 3.6: Ejemplo de JSD tomado de Dix et. al (1993)

Para (Curry y Monk, 1990b) este método presenta las siguientes ventajas:

• Permite algún tipo de análisis cognitivo, en la medida que es posible puntuar la complejidad de los diálogos en función de las opciones que hay en él.

• Es muy usado y conocido por lo que los diseñadores no necesitarían llevar a cabo grandes esfuerzos adicionales para aprender a utilizarlo para este fin.

• Es explícito en cuanto a todas las acciones que pueden ser ejecutadas con un objeto.

• Existen herramientas informáticas que podrían ayudar a su uso.

En cuanto a sus desventajas encuentran que no es muy apropiado para representar la concurrencia y que aunque puede ser adecuado para el análisis de tareas, el propósito de especificación de diálogos estaría difícilmente cubierto.

3.2.4.3.1.2. Diagrama de Transición de Estados (State Transition Networks ,STN).

En un diagrama de transición de estados los diálogos son representados de la siguiente manera: los círculos indican estados en los que el sistema puede encontrarse en un momento dado, y las flechas transiciones desde un estado a otro. Estas transiciones están etiquetadas con las acciones del usuario que dispararon la acción del sistema y la respuesta o feed-back que el sistema proporciona. En la figura 3.7 se muestra un ejemplo de como podría representarse un diálogo por medio de un red de transición de estados encargado de controlar la selección de un menú de una herramienta (círculo o línea), y los distintos pasos necesarios para su especificación.

En esta descripción cada círculo es un estado en el que el sistema puede encontrarse. Entre los estados están las transiciones, identificados por flechas. En la parte de arriba de la flecha está la acción del usuario que desencadena la respuesta del sistema indicada en la parte de abajo de ésta. Por ejemplo, para dibujar un círculo el usuario seleccionaría la herramienta círculo en el menú. A continuación señalaría el punto central del círculo y después un punto cualquiera de la circunferencia. En este momento, el sistema dibujaría el círculo y la línea de diálogo se cerraría. Los pasos para dibujar una línea serían similares, con el añadido de existir un bucle que permitiría realizar varias lineas.

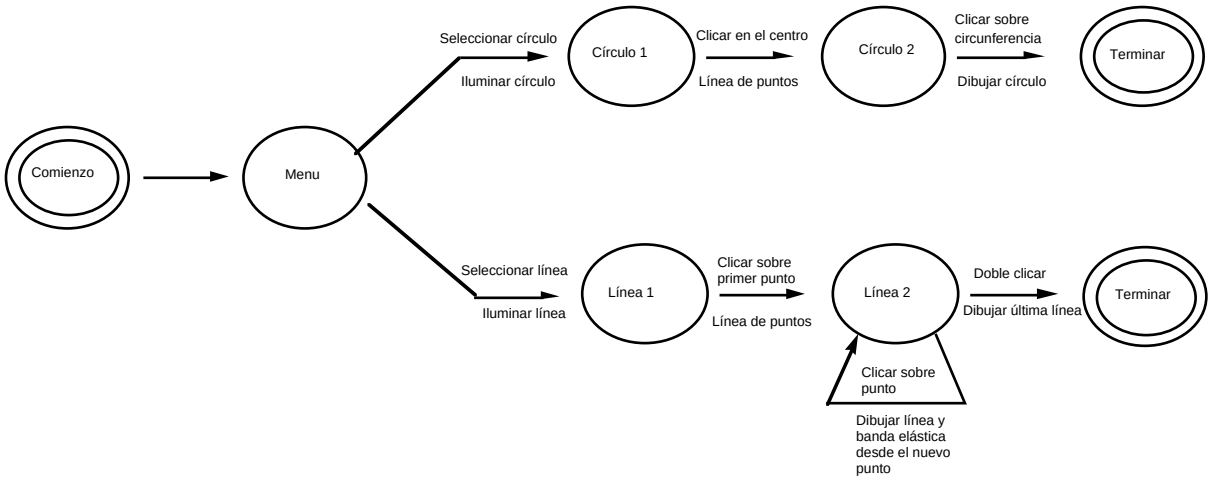


Figura 3.7: Descripción STN de un diálogo que controla el dibujado de una línea o un círculo a través de un menú. Tomado de (Dix, et al., 1993).

Jacob (Jacob, 1983; Jacob, 1986) comenta que las redes de transición de estados presentan un gran poder formal con la ventaja de ser más intuitivas y comprensibles.

Ha habido varias extensiones a las STN más básicas, entre ellas están:

• Redes de transición recursivas: Una red puede hacer una llamada a otra red tal y como si se tratara de una subrutina. Esto permite especificar un sistema complejo como un conjunto de otros más simples.

• Redes de transición aumentada: Si una descripción de un diálogo (Wasserman, 1986) tuviera que especificar todos los mini-cambios asociados a cada acción del usuario la notación sería excesivamente detallada y por tanto inútil. Imaginemos si tuviéramos que especificar el movimiento del ratón en una dirección dada pixel por pixel. Por ello, las especificaciones necesitan aludir al concepto de buffer, el cual almacena una cierta cantidad de interacción dentro de un estado, mostrando el progreso del diálogo en función del contexto en que ocurren los sucesos.

En cuanto a las ventajas y desventajas de estos modelos comentar en primer lugar que los STN simples son menos usados que algunas de sus variantes, por lo que los problemas que se les puede achacar están en ocasiones superados por otras aproximaciones.

Entre las ventajas (Curry y Monk, 1990b) señalan:

• Son bastante fáciles de entender al depender de las habilidades visuales de los diseñadores.

• Al describir un diagrama de transición de estados aparece de una manera automática una estructura de nodo o encrucijada, la cual hace referencia a lugares o puntos en los que el usuario tendrá que decidir cuál es el camino correcto a seguir. Este concepto es intuitivamente muy plausible para los diseñadores, ya que muy a menudo se encuentran razonando sobre situaciones parecidas.

Entre los inconvenientes se encuentran:

• Al especificar diálogos en los que son posibles combinaciones de diversas opciones se produce una explosión combinatoria de transiciones que convierte los diagramas en inmanejables. Esta situación es particularmente inconveniente al especificar diálogos concurrentes.

• Escapes y ayudas. Los escapes son acciones que permiten salir de un estado sin haber llegado al final del arco correspondiente. Especificar esa posibilidad mediante STN añade un arco a cada estado que llevaría al menú principal, lo cual complicaría enormemente la descripción del sistema. Las ayudas son una complicación semejante.

3.2.4.3.1.3. State Charts (SC) de Harel

SCes una extensión de STN, que extiende sus facilidades y le añade algunas modificaciones. Los constructos que tiene SC (Curry y Monk, 1990b) son:

• Jerarquía: muchos estados de orden bajo pueden ser agrupados en un estado de orden superior.

• Ortogonalidad: Un estado puede ser igual a la combinación de cualquier estado activo dentro de dos grupos de estados. De ese modo se reduce enormemente la explosión combinatoria.

• Sucesos: Los sucesos que conectan estados pueden ser etiquetados también como condiciones-acciones, las cuales pueden producir reacciones en cadena en otras partes del diagrama.

• Estado por defecto: una flecha indica el subestado por defecto en que entra el sistema cuando entra en un estado compuesto de subestados.

• Historia: el sistema es capaz de recordar en donde se encontraba cuando otro estado ocurrió. Un ejemplo es cuando un televisor recuerda en que canal estaba puesto cuando fue apagado.

Otra ventaja importante es que existe una herramienta informática que permite el análisis de los diálogos especificados mediante esta notación. No obstante, esta herramienta es sumamente cara así que su uso no está ampliamente difundido. Curry et. al. (Curry y Monk, 1990b) le otorgan la puntuación más alta de todos los modelos que revisan.

3.2.4.3.1.4. Generalised Transition Networks (GLTN) de Kieras y Polson.

En la sección dedicada al modelado del usuario veremos una notación denominada CCT de Kieras y Polson, el cual es un método basado en GOMS. En sus primeras formulaciones, no obstante, el sistema original de Kieras y Polson se componía de dos partes. Una de ellas era una descripción del diálogo usando una notación semejante a la de STN llamada GLTN y la otra era la que modelizaba al usuario mediante una serie de reglas de producción. El objetivo era construir un sistema informático alimentado por esas dos notaciones que simulara el comportamiento de ambos sistemas y diagnosticara los puntos en los que hubiera desajustes (Kieras y Polson, 1985).

No obstante, el trabajo posterior de Kieras y Polson se han centrado ante todo en la parte correspondiente a las reglas de producción y han dejado de prestar atención a GLTN.

3.2.4.3.1.5. Petri nets orientadas al objeto

Las redes de Petri son un formalismo utilizado para la especificación de sistemas computacionales concurrentes asíncronos (Peterson, 1977). Así, su uso principal ha sido la del modelado de sistemas de eventos en los cuales es posible la concurrencia , pero existen limitaciones sobre la precedencia o frecuencia de estas ocurrencias. Además, es un formalismo utilizado también para la descripción de sistemas expertos (Agarwal, 1992)

Una red de Petri puede ser descrita del siguiente modo. En un gráfico de red de Petri hay dos tipos de nodos: círculos (llamados lugares) y barras (llamadas

transiciones). Estos nodos están conectados por medio de arcos dirigidos de los lugares a las transiciones y de las transiciones a los lugares. Si un arco se dirige desde el nodo i hasta el nodo j, entonces i es un input a j y j es un output de i.

Además de esta descripción estática, una Petri net tiene una serie de propiedades que resultan de su ejecución. Suponiendo que el gráfico tiene una marca (generalmente un punto negro) en el estado P1 entonces nos encontramos con una Petri net marcada. Estos puntos negros pueden ser movidos de una manera similar a fichas en un juego de tablero, como consecuencia del disparo de una serie de transiciones sobre la red. Cada disparo es producido pasando una marca negra a todos los outputs conectados con un nodo siempre y cuando las transiciones intermedias estén preparadas. Una transición intermedia está preparada si todos los lugares conectados con ella son disparados simultaneamente. En la figura 3.8 es posible ver el efecto que tendría el disparo de las reglas asociadas a P1 y a P5 asumiendo que estuvieran marcados con un punto. Dado que P1 se conecta a través de la transición 2 con P2 y P3 y sólo una condición es necesaria para disparar esta transición ambas pasan a tener un punto. En cambio, P5 no produce ningún efecto ya que la transición 3, con la que está conectado, necesitaría de la condición P3 siendo disparada simultaneamente.

Este tipo de notación podría utilizarse adjudicando significados en relación con un diálogo hombre-ordenador a los distintos puntos.

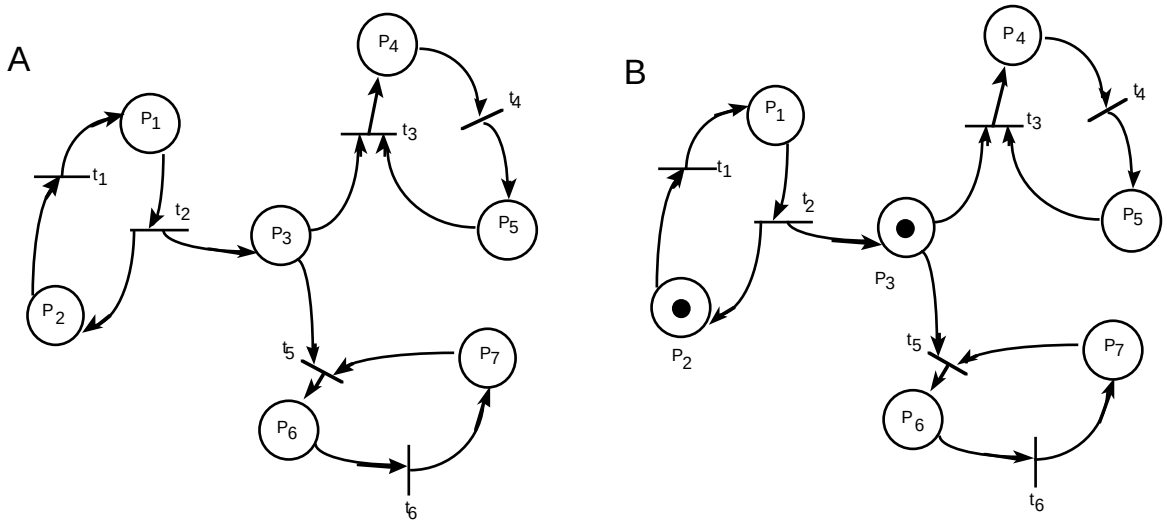


Figura 3.8: Gráfico Petri net Tomado de Peterson (Peterson, 1977).

El uso de las Petri nets aplicado a la descripción de diálogos puede ser considerado como relativamente reciente. Así, Van Biljon (Van Biljon, 1988)

introduce una serie de modificaciones dirigidas a manejar la complejidad derivada de su uso a la hora de aplicarlas a la descripción de diálogos. Estas son: la posibilidad de anidar Petri nets, simplificaciones del modelado del input-output y etiquetado de transiciones correspondientes a las condiciones de error y metafunciones.

Curry y Monk (Curry y Monk, 1990b) comentan la existencia de PNO (Objetos Petri Net) en donde el flujo de control es externo al objeto la mayor parte del tiempo, aunque en ocasiones será necesaria la descripción interna de los objetos. Los objetos son descritos tanto en términos de sus atributos como de la conducta de interfaz que tienen. Como resumen acaban haciéndole los siguientes comentarios:

1. Es un método muy bien respaldado por sus usos en otras áreas. Es gráfico, preciso y capaz de expresar concurrencia.

2. No es capaz de distinguir fácilmente entre módulos.

3. No se beneficia del diseño top-down.

4. No es muy intuitivo, ya que resulta difícil captar la dinámica del interfaz utilizando sólo esta representación.

3.2.4.3.2. Basadas en sucesos.

Los diagramas basados en sucesos son en cierto modo complementarios a los basados en estados. La información de aquellos podría ser traspasada a este tipo de notación.

3.2.4.3.2.1. Event-Decomposition Graphs (EDG´s) de Chakravarty y Klein

En este notación se describe el sistema en función de las posibles acciones que los usuarios pueden llevar a cabo. Los brazos de los nodos indican los estados en los que el sistema puede irse encontrando, mientras que los nodos indican posibles acciones que el usuario podría llevar a cabo. Es pues una representación complementaria a los sistemas basados en eventos, en los que los brazos indican sucesos y los nodos estados.

EDG está basado sobre grafos AND/OR. En un nodo AND, todos los sucesos deben ser verdaderos para lograr llevar a cabo una tarea concreta. En un nodo OR

sólo uno de los sucesos necesita ser verdadero. En el siguiente ejemplo se muestra cómo dibujar una línea mediante el uso del ratón en un programa de aplicación de gráficos.

En la notación, los términos en cursiva indican acciones del usuario y los que usan un texto normal indican acciones o respuestas del sistema. Para dibujar la línea es necesario en primer lugar apretar el botón del ratón sobre algún punto, momento en el que el sistema registra esta acción y además realizar la acción de extender esta línea. Una vez hecho es posible bien terminar directamente (con lo que la línea dibujada sería sólo un punto) o mover el ratón y apretar el botón de nuevo. Después de eso podría bien repetir la secuencia extender línea para dibujar un número indefinido de nuevas lineas o borrar la última línea. En cualquier caso, la opción de terminar aparecerá y se podrá cerrar la interacción y dar por terminado el dibujo.

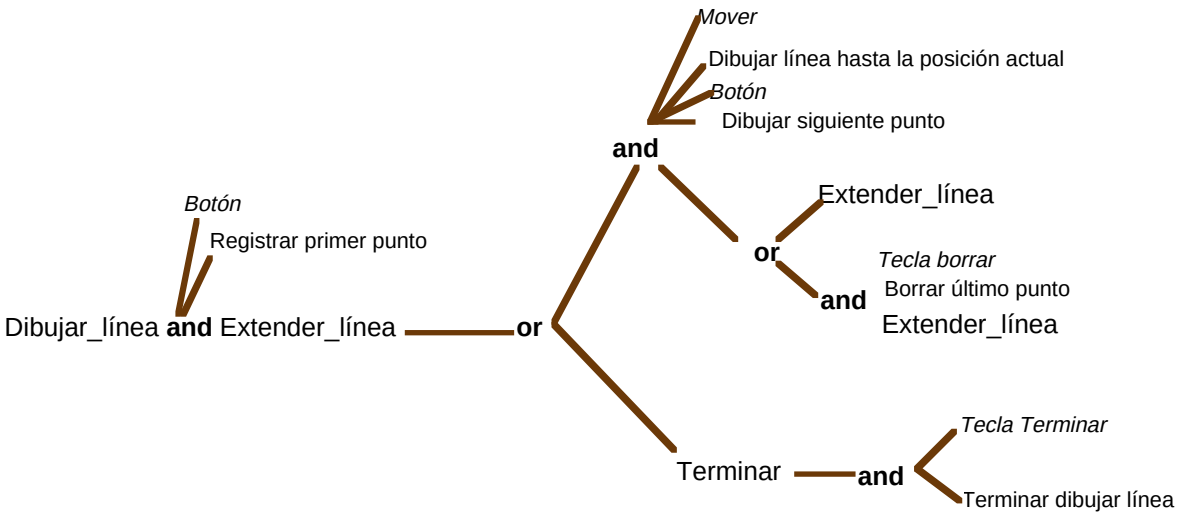


Figura 3.9: Ejemplo de gráfico de descomposición de sucesos tomado de Curry et. al (1990).

Este sistema presenta algunas ventajas potenciales (Curry y Monk, 1990b):

• No es necesario especificar todos los componentes de una descripción EDG en el primer momento. Procesos de sucesos más complejos pueden ser analizados en sucesivos pasos.

• La estructura de árbol es bastante familiar, y sería relativamente fácil crear herramientas informáticas o aprovechar otras existentes para este fin.

• No es necesario especificar cada uno de los componentes sencillos de un estado, sino sólo aquellos que contribuyen a la funcionalidad.

En cuanto a las desventajas:

• No permite manipular la concurrencia. En un sistema basado en sucesos, no es posible que dos sucesos ocurran simultaneamente.

• No es muy efectiva a la hora de manipular interrupciones en el diálogo.

3.2.4.4. Aproximaciones formales/algebraicas

En esta sección se pueden incluir algunos métodos formales de propósito general que han sido adaptados al modelado de diálogos (tal y como Z o VDM) y otros lenguajes especialmente desarrollados para el modelado de diálogos. Veremos este último tipo de notaciones más cuidadosamente a continuación.

3.2.4.4.1. Lenguajes especiales desarrollados para el modelado de diálogos.

El modelo más conocido es CSP de Alexander

3.2.4.4.2. Communicating Sequential Processes (CSP) de Alexander

El siguiente texto estará basado en (Alexander, 1990; Alexander, 1987), aunque sigue más fielmente el primero de ellos.

Alexander argumenta en primer lugar que es necesario proponer métodos que permitan a los programadores separar los distintos módulos de programación con objeto de conseguir mayor nitidez y claridad en los aspectos de funcionamiento, permitiendo realizar modificaciones posteriores fácil y rápidamente. El interfaz de usuario es uno de los módulos separables. CSP es un lenguaje matemático que permite llevar a cabo esa especificación y, además, dado que es ejecutable, permite a los diseñadores experimentar con los interfaces que crean para de ese modo detectar los errores que pudieran producirse.

Sistemas de especificación anteriores a CSP resultaban adecuados para los interfaces basados en lenguajes de comandos. Así, la notación denominada BNF (Backus Naus Form) y los diagramas de transición de estados, al haber sido diseñados para especificar lenguajes de programación, podrían adaptarse para

tratar con los lenguajes de comandos, los cuales podrían considerarse subconjuntos simplificados de aquellos.

No obstante, existen dos problemas que las notaciones anteriores no han sido capaces de afrontar. En primer lugar, no son ejecutables, por lo que no pueden producir prototipos que demuestren el funcionamiento deseado inmediatamente y, en segundo lugar, no son capaces de afrontar los nuevos sistemas de diálogo (manipulación directa). En concreto, no pueden manipular ni la concurrencia ni la presentación en pantalla.

Otros métodos, tales como las redes de Petri, sí que son capaces de manejar esas situaciones. No obstante, Alexander considera que CSP presenta ventajas sobre todos los otros métodos.

CSP distingue entre acontecimientos y procesos, en donde cada proceso es disparado por un acontecimiento. Originalmente concebido como una extensión del lenguaje Algol, su objetivo es describir la conducta de sistemas que son vistos como grupos de procesos comunicándose en un entorno.

En realidad, el CSP original no es el utilizado para llevar a cabo estas descripciones. La versión de Alexander se denomina eventCSP y así será llamada a partir de ahora.

En breves palabras, el formalismo es como sigue.

```
Un proceso es definido por una sentencia de esta forma:

<Nombre de proceso>=<Expresión de proceso>

Si e es un suceso y P y Q son procesos, entonces lo siguiente también
son procesos:
(e -> P) : Indica secuencia. Sucede e y entonces se comporta como P.
P Q : Elección. Comportarse como el proceso P o como el proceso Q.
P ; Q : Secuencia. Comportarse como P y entonces como Q.
P || Q: Paralelo. Comportarse como P y a la vez como Q sincronizando
los acontecimientos comunes.
SKIP: terminación. El proceso ha sido completado con éxito.
```

No obstante, eventCSP es meramente una de las partes del formalismo, dirigida principalmente a representar las conexiones entre los distintos procesos y la sucesión de sucesos que van poniéndolos en marcha. Así, existen otras herramientas, denominadas conjuntamente SPI (Simulating and Prototyping Interaction), que presentan las siguientes funciones: a) Llevar a cabo una

simulación/ejecución del modelo representado en eventCSP (SPIOutliner) b) Llevar a cabo una simulación/ejecución que muestra los efectos en la pantalla de las distintas interacciones (SPIScenario) y c) Describir exactamente cada uno de los acontecimientos de modo que se determine con claridad las consecuencias de su puesta en marcha (eventISL-Interaction Specification Language).

En conclusión, CSP y SPI permiten conjuntamente llevar a cabo una simulación de las representaciones y manejar la concurrencia. No obstante, el otro gran problema, el manejo de la pantalla no está completamente solucionado, ya que no hay manera de comprobar la consistencia entre la presentación en pantalla simulada y el funcionamiento del modelo, ya que no se trata más que de una serie de gráficos conectados a los distintos puntos de la representación. Otro problema, no obstante, se halla en el que la decisión acerca de la granulosidad del modelo depende del diseñador, y que dependiendo de este las representaciones alcanzan tamaños excesivos.

No obstante, Alexander opina que esta representación es altamente adecuada, ya que combina un cierto grado de formalismo que permite realizar pruebas formales del sistema y su habilidad para llevar a cabo prototipos lo cual permite la exploración y el uso de la intuición.

Para Curry et. al (Curry y Monk, 1990b) los defectos de esta notación son los siguientes:

1. Los diseñadores tienen problemas en elegir un nivel de abstracción apropiado para una especificación. Los eventos pueden ser un simple suceso o algo muy complejo. El diseñador tiene que decidir que aspectos son de interés antes de que la especificación se realice.

2. Necesita mucha precisión. Aquellos que la utilicen deben conocerla muy bien para derivar los beneficios que se resultan de su uso.

3. No maneja la presentación en pantalla.

3.3. Modelos formales del sistema

3.3.1. Introducción

En este apartado veremos una presentación de tres tipos de modelos formales del sistema. En primer lugar veremos el modelo PiE que puede considerarse como uno de los pioneros en esta línea. En segundo lugar veremos el modelo de agente de Abowd, como un progreso realizado sobre esos esfuerzos iniciales. Finalmente, veremos otros modelos de agente que han sido propuestos con el objeto de servir de base a sistemas UIMS (Sistemas de gestión del interfaz de usuario).

3.3.2. El modelo PiE de Dix

Dix (Dix, 1991) propone realizar una descripción del sistema desde el punto de vista del usuario, sin hacer referencia a los estados internos del sistema. Esta descripción es interesante para un diseñador porque le permite conocer cual será el modelo mental que se formará el usuario del sistema que está utilizando, ya que el usuario sólo tiene acceso a la información externa, no a la información interna. Por ello, este modelo recibe también el nombre de modelo de caja negra (black box). Esta noción de caja negra es, según Abowd (Abowd, 1991), bastante común en ingeniería de software y desarrollo de sistemas, apareciendo con el nombre de abstracción de datos o tipo de datos abstractos. El ejemplo más sencillo de modelo PiE es el mostrado en la figura 3.10:

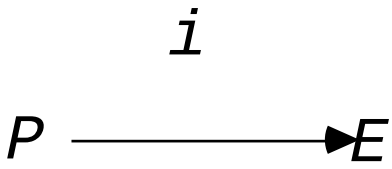


Figura 3.10: Esquema de modelo PiE.

En ese esquema, P hace referencia al conjunto de comandos que dispone el usuario, E es el espacio de efectos que el usuario puede obtener e i es la computación llevada a cabo por el sistema.

Este modelo es muy simple y su objetivo es sobre todo mostrar el concepto de "caja negra". Sin embargo, para llevar a cabo análisis de principios interactivos del modo en que Dix se propone, es necesario refinamientos posteriores. Veamos a

continuación el análisis que hace del principio de predictibilidad. El lector interesado puede consultar otros principios en las siguientes referencias (Abowd, 1991; Dix, 1991; Dix, et al., 1993; Harrison y Abowd, 1991; Harrison y Timbleby, 1990a; Johnson y Harrison, 1990).

3.3.2.1. El análisis de la predictibilidad

En la figura 3.11 se muestra un refinamiento del modelo PiE que lo hace apropiado para tratar la propiedad de predictibilidad, la cual es un concepto relacionado cercanamente al de WYSIWYG (What you see is what you get: Lo que ves es lo que tienes). Este principio, a menudo reclamado como propio en muchos productos comerciales, es objeto de un análisis detallado por Dix en varias ocasiones (Dix, 1991; Dix, et al., 1993) que muestra las limitaciones formales de su verdadera aplicabilidad.

En ella, se puede ver como añadido con respecto a la figura 3.10 la distinción en el efecto entre el resultado (el punto final de nuestra interacción con el ordenador que en muchos programas coincide con la versión impresa) y la pantalla. Esta distinción es el punto inicial de la discusión del principio de WYSIWYG por medio del modelo PiE.

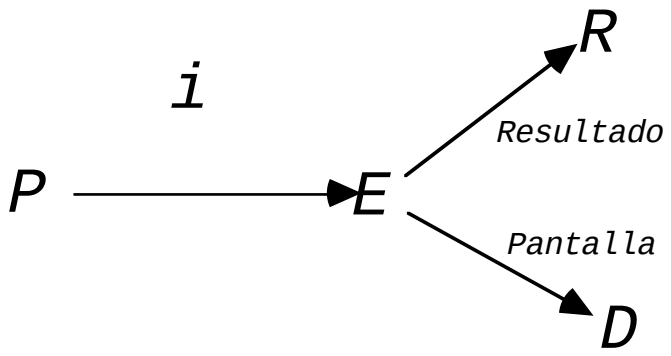


Figura 3.11: Modelo PiE refinado.

La predictibilidad hace referencia a una propiedad del sistema que permitiría predecir el resultado de un comando a partir de la información que existe en la pantalla, asumiendo que el usuario no conoce o ha olvidado la historia de

interacción anterior[3]. Existen dos formas interesantes de entender la predictibilidad que pueden establecerse a partir del modelo PiE refinado.

• Predictibilidad del resultado. Según este principio, existiría una función trasparente tal que nos permitiría pasar del aspecto de la pantalla al resultado que hemos obtenido. Es decir, si lo que vemos es lo que hemos conseguido.

• Predictibilidad del efecto. Este es un principio muy poderoso. Este dice que no hay nada en el estado del sistema E que no pueda ser predicho a partir del aspecto de la pantalla D.

Obviamente, estos dos principios son demasiado rigurosos y probablemente poco realistas. Normalmente, la mayoría de los programas trabajan con documentos e información que no puede ser totalmente representada en la pantalla. Por ejemplo, un procesador de textos manejará documentos que abarcarán más que la amplitud de cualquier pantalla concebible. No obstante, el usuario podrá mover el texto hacia arriba o hacia abajo para ver el resto del documento. La posibilidad de examinar el estado del sistema se denomina estrategia y a partir de ella es posible definir una vista del sistema más amplia que la basada exclusivamente en la pantalla. Esta vista se denominaría efecto observable.

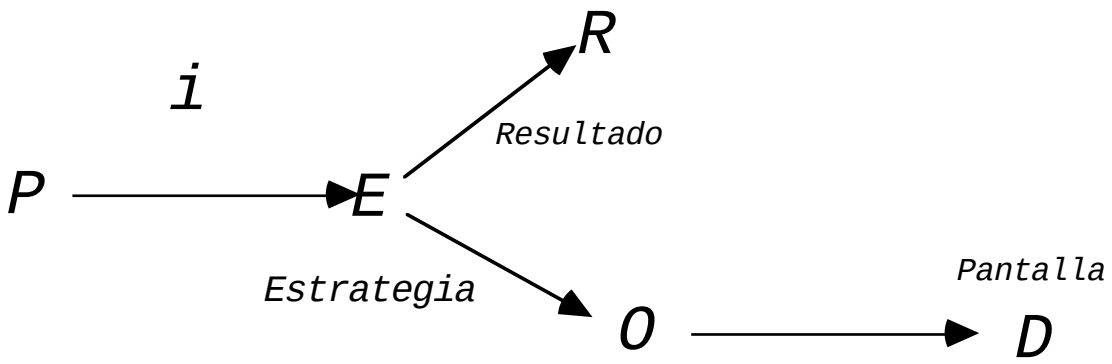


Figura 3.12: Modelo PiE con efecto observable.

[3] ...debido a que ha ido a tomar una taza de té y no recuerda como dejó su trabajo. Este problema es denominado por Dix "gone away for a cup of tea problem".

Este efecto observable permite reescribir los principios anteriormente descritos de una manera más realista:

• Predictibilidad del resultado. Según este, existiría una función trasparente tal que nos permitiría pasar de efecto observable al resultado que hemos obtenido. Es decir, si lo que vemos y/o podemos ver a través de una estrategia de exploración es lo que hemos conseguido.

• Predictibilidad del efecto. Este principio diría que uno puede observar el estado completo del sistema utilizando las estrategias adecuadas.

Como resultado final de esta discusión, vemos que el principio de predictibilidad (uno de los componentes del concepto de WYSIWYG) tiene un alcance relativamente limitado, una vez analizado desde un punto de vista formal. Siempre habrá efectos del sistema que no podrán ser predichos sin cierta cantidad de exploración, por lo que los diseñadores deberán ser conscientes de la necesidad de ofrecer métodos adecuados para llevarla a cabo. Ello nos llevaría al concepto de esfuerzo adecuado (habría que proporcionar medios para que la exploración no fuera demasiado costosa) y permitir estrategias pasivas de exploración (estrategias que permitieran examinar el estado del sistema sin necesidad de modificarlo).

3.3.2.2. Críticas al modelo PiE

Abowd (Abowd, 1991) comenta que este modelo ha sido criticado desde dos puntos de vista:

Desde el punto de vista de los ingenieros de software, estas técnicas no son constructivas, es decir, no proporcionan suficiente soporte para los diseñadores que intentan ir desde una especificación inicial hasta la implementación. En cierto modo, un modelo de caja negra en realidad se aleja de los problemas de implementación por lo que su utilidad desde ese punto de vista es muy dudosa. Por otro lado, aunque teóricamente sería posible utilizarlo para hacer pruebas formales de determinadas propiedades en sistemas reales, debido a que no intenta hacer ningún tipo de modularización, llevar a cabo esas pruebas sería interminable.

Desde el punto de vista de la psicología, se dice que es muy dudoso que sea posible especificar un conjunto de principios formales que, una vez demostrados su

cumplimiento en un sistema concreto, garanticen la usabilidad de éste. Esta crítica sugiere que en cuestiones de diseño nunca sabremos suficiente como para anticipar todos los problemas, y, por tanto, siempre será necesario llevar a cabo ciclos posteriores de evaluación empírica por medio de prototipos.

3.3.3. El modelo de agente de Abowd

Para Abowd (Abowd, 1991) el modelo de agente viene a llenar un hueco en las notaciones propuestas por otros autores, ya que, aun siendo una notación formal, permite la introducción de conocimientos psicológicos dentro de ella. De este modo, supera los problemas de todas las otras notaciones, ya que, para él, las notaciones basadas psicológicamente poseen problemas conceptuales y notacionales, mientras que aquellas con validez formal no poseen ningún tipo de validez psicológica.

La mayoría de los sistemas propuestos y utilizados en la comunidad de investigadores en HCI son adecuados sobre todo como técnicas post hoc de análisis de diseños. El modelo que ellos proponen intenta ser útil a la hora de definir el diseño, siendo por tanto no sólo evaluativo sino también prescriptivo.

3.3.3.1. Tipos de datos abstractos (abstract data type: ADT)

Un sistema puede ser especificado como un estado con un conjunto de objetos, y una serie de operaciones posibles sobre esos objetos. Esto nos lleva a la definición de tipo de dato abstracto. Un tipo de dato abstracto puede ser especificado indicando los estados en los que se puede encontrar, las operaciones que le afecten y, generalmente, una serie de axiomas que indican las constricciones (axiomas o ecuaciones) que afectan a los posibles estados del tipo de datos.

Por ejemplo, dentro de un procesador de textos podrían encontrarse un gran número de tipos de datos abstractos combinados. Uno de ellos, es por ejemplo, el tipo posición del cursor. En la figura 3.13 es posible ver una definición de este tipo de datos abstractos (tomado de Harrison & Abowd, 1991).

```
ADT posición
STATE posición
OPERATIONS
 origen: -> posición
 xorigen: posición -> posición
 yorigen: posición -> posición
 arriba: posición -> posición
 abajo: posición -> posición
 derecha: posición -> posición
 izquierda: posición -> posición

AXIOMAS
 FOR todo p IN posicion
      arriba(yorigen (p))= yorigen (p)
      izquierda (xorigen(p)) = xorigen (p)
      xorigen(yorigen (p))= yorigen(xorigen(p)) = origen
      derecha (izquierda(p))=p
      arriba (abajo (p))=p
END  ADT posición
```

Figura 3.13: Una descripción del tipo de dato abstracto posición del cursor en un procesador de texto.

En esta descripción vemos que existe un estado en el que el cursor tiene una posición dada, y que sobre la posición de ese cursor es posible realizar una serie de operaciones que toman un parámetro inicial y devuelven un resultado que es el nuevo estado del cursor (izquierda por ejemplo tomaría una posición y devolvería una posición). Esas operaciones no obstante estarían limitadas por una serie de axiomas, como, por ejemplo, que la operación izquierda cuando la posición del cursor está en el origen de las x tiene como resultado la misma posición en la que estaba (el cursor no podría ir más allá del margen izquierdo).

3.3.3.2. Una definición de agente

Aunque los ADT son una parte importante de un agente, éste involucra otros elementos. En su definición de agente, Harrison y Abowd (1991) intentan incorporar dos principios arquitecturales que es necesario señalar: En primer lugar facilidad de expresión, lo cual hace referencia a que la notación es capaz de sugerir la alternativa más correcta o adecuada cuando un agente podría ser descrito de muchas maneras distintas. En segundo lugar, composicionalidad, lo cual hace referencia a que un agente complejo puede ser descrito en términos de agentes más simples. Así, la tarea de describir un agente complejo puede ser modularizada y la tarea de formular propiedades y luego realizar pruebas formales puede ser estructurada.

Hay tres aspectos de un agente que es necesario describir:

• El estado interno del agente: Es la información que persiste a lo largo de la vida del agente, pero que es privada a éste. Esta información es en buena parte una "caja negra" debido a que no estamos interesados en el potencialmente infinito nivel de detalle que subyace a cada agente. Sólo un conocimiento a nivel de valores y conexiones con variables potencialmente relevantes es necesario.

• Comunicación: Los agentes se comunican entre sí por medio de mensajes. Estos mensajes pasan por canales de input-output explícitos que están restringidos del siguiente modo: sólo un agente puede enviar mensajes a través de ellos en cada momento y sólo un agente puede recibirlos. Un mensaje pasado a lo largo de un canal se convierte en un suceso que provoca una transición de estados en el agente receptor.

• Conducta: La conducta de un agente es la descripción de los sucesos en los que el agente puede participar, así como las transiciones de estado que se producen como consecuencia de la ocurrencia de esos sucesos. Este punto es de gran interés para Harrison y Abowd, ya que ello supone tener dos visiones coordinadas del agente, la basada en estados y la basada en sucesos. La visión basada en estados describe las operaciones que pueden ser ejecutadas sobre un agente y el efecto que estas tienen sobre el estado interno. La visión basada en sucesos describe qué secuencias de sucesos son permitidos por uno o más agentes en un grupo. La principal aportación del modelo de agente es proporcionar un método para unificar estas dos visiones.

A continuación se muestra una definición de agente de un documento de texto tal y como podría encontrarse en un procesador de textos orientado a la pantalla. Esta descripción se apoya en los conceptos introducidos en la definición de ADT, pero los amplia para integrar las definiciones de sucesos.

3.3.3.2.1. Pruebas de principios de interacción.

Harrison et. al (Harrison y Abowd, 1991) ofrece un análisis del principio de predictibilidad en un paquete gráfico. Este análisis muestra cómo intentar seleccionar un objeto gráfico situado sobre otros objetos gráficos no es una tarea sencilla debido a que tres jerarquías se solapan en esa estructura: Una jerarquía visual (el solapamiento existente entre los objetos), una estructural (la forma en que los objetos fueron agrupados) y una temporal (el momento en que fueron

```
AGENTE: documento
ESTADO: documento
USES caracter, position 'utiliza otros agentes menos
complejos.
ALFABETO 'detalla  el alfabeto para el agente en
términos de    sucesos y canales de input y
output.Borrar e  insertar son sucesos e inputs que
provienen del teclado. Presionado, suelto y posición
son sucesos e inputs que provienen del ratón.
  SUCESOS borrar, presionado, suelto
  INPUTS insertar, posición
  OUTPUTS pantalla
CONDUCTA 'define  los procesos en los que puede
involucrarse el sistema,  los sucesos que los ponen
en marcha y los sucesos a los que da lugar.
DOC = borrar -> DOC
      | insertar ? ch -> DOC
      | presionado ->  MOVER
MOVER = posición ? <x,y> ->MOVER
      | suelto -> DOC
'sólo se definen dos procesos. En el primero      el
sistema está  esperando que ocurra alguno de los
sucesos definidos: borrar un caracter,  insertar un
caracter o mover el cursor. Los dos primeros sucesos
terminan encadenandose con el proceso original  DOC.
El tercero da lugar al  proceso MOVER. Este proceso
no  finaliza mientras no se suelte el botón del
ratón. Cuando este proceso es inactivado, el proceso
DOC comienza de nuevo.
OPERACIONES 'Esto sigue la estructura de un ADT.
nuevo: ->documento
insertar: caracter documento ->documento
borrar : documento -> documento
marcar: posición documento -> documento
señalar : posición documento -> documento
deshacer : documento -> documento
CONEXION ENTRE SUCESOS 'define la conexión entre las
operaciones que  el usuario puede realizar y los
inputs que el sistema puede recibir.
  borrar |--> borrar
  insertar.ch |--> insertar (ch)
  presionado, posición <x,y>  |--> señalar (contexto
((estructura, (x,  y))) 'cuando se aprieta el ratón
se señala   una posición dentro de un contexto
definido en otro lugar y que básicamente consiste en
el texto sobre el que se está trabajando.
posición <x,y>, suelto |   --> marcar ((estructura,
contexto (x,y)))'hay  que tener en cuenta que para
poder soltar  el botón es necesario que estuviera
apretado, así este suceso da lugar a la selección de
un texto.
……
END AGENTE documento
```

Figura 3.14: Descripción de agente de un documento en un procesador de textos orientado a la pantalla.

creados). Un análisis formal de esta situación lleva a mostrar que sólo teniendo un concepto de memoria explícito en el usuario es posible mantener el principio de predictibilidad. Y esa memoria requeriría un modelado cognitivo de modo que fuera posible apreciar propiedades psicológicas que ayudarán a determinar la descripción del sistema (por ejemplo, utilizar una jerarquía visual es mejor que una temporal o estructural).

3.3.4. Otros modelos de agente

El modelo PiE de Dix ha sido criticado por estar demasiado alejado de los problemas concretos de los ingenieros de software que intentan llevar a cabo diseños reales (Abowd, 1991). Las técnicas basadas en el modelo de caja negra no son de gran ayuda para la especificación de sistemas concretos, estando su dominio de acción en el análisis de principios abstractos que iluminen cuestiones generales acerca del diseño.

Los siguientes modelos son movimientos hechos con la intención de proporcionar a los diseñadores con sistemas denominados UIMS (User Interface Management Systems), los cuales proporcionen soporte a la tarea de llevar a cabo diseños concretos. Para ello, estos modelos proporcionan generalmente un modelo del reparto de tareas entre diversos componentes de un sistema interactivo y un método para la integración de estos.

En general, la mayoría de estas propuestas hacen uso de la metáfora de la programación orientada al objeto, postulando diversos componentes en cada objeto que corresponden a las entidades consideradas relevantes en cuanto al objetivo de diseñar buenos interfaces hombre-computador.

Dos ejemplos que hoy podrían considerarse históricos, (a pesar de que en realidad continúan estando muy presentes en una gran variedad de situaciones), son (Dix, et al., 1993): el modelo Seeheim y el paradigma MVC (Model, View, Controller) del entorno de programación Smalltalk.

• El modelo Seeheim: Ya comentado anteriormente en el apartado dedicado a notaciones de descripción de diálogos, este modelo distinguía entre tres componentes de un UIMS: presentación (responsable de la apariencia del interface y del input y el output disponible al usuario), control del diálogo (responsable de la comunicación entre la presentación y la aplicación) y la aplicación (componente encargado de la funcionalidad real). Esta es una distinción paralela a la existente

entre las capas léxica, sintáctica y semántica en los compiladores. Un problema de este modelo es que no muestra como crear sistemas grandes y complejos a partir de módulos más pequeños.

• El modelo MVC: Este modelo sí presenta una aproximación basada en bloques al estar conectado con Smalltalk, uno del primeros sistemas de programación orientada al objeto con éxito. En él, la conexión entre la funcionalidad de la aplicación y la presentación puede ser construida mediante la triada Modelo (funcionalidad de la aplicación), Vista (output gráfico o textual) y Controlador (input al sistema). El comportamiento básico de los MVC está incorporado dentro de clases de objetos que pueden ser ejemplificados y modificados a la hora de especificar un sistema concreto.

Coutaz (Coutaz, et al., 1993a) distingue entre modelos arquitectónicos conceptuales y modelos arquitectónicos implementacionales. En los primeros, cada entidad denota un concepto (por ejemplo, la noción de control del diálogo), mostrando el modelo como estos conceptos se relacionan entre sí. En los modelos implementacionales, una entidad se corresponde con un componente de software (por ejemplo, el controlador del diálogo que sería una pieza específica de software encargado de esta función), mostrando los modelos las relaciones entre estos componentes. Existe en estos últimos, por tanto, una cercanía mayor a cuestiones prácticas en ingeniería de software tales como protocolos de comunicaciones y conexiones con código reusable (p.e. bibliotecas de software). Aquí no llevaremos a cabo una revisión completa de los modelos que podrían incluirse dentro de estos apartados. Únicamente veremos una introducción a los conceptos básicos de tres de estos modelos (PAC, GIO y MVC) y cómo estos se relacionan entre sí.

3.3.4.1. Conceptos básicos en arquitecturas conceptuales: El concepto de agente, de interactor y su relación con el de objeto

Un agente es un sistema de procesamiento de la información completo que incluye receptores de sucesos, transmisores de sucesos, una memoria que mantiene estados, y un procesador que cíclicamente maneja estas entidades. Ahora bien, aunque en esta definición es posible incluir tanto agentes cognitivos (con capacidad de satisfacer sus propios objetivos), como agentes reactivos, (que no poseen sus propios objetivos sino sólo aquellos codificados explícitamente por el diseñador humano) los modelos tenidos en cuenta aquí sólo consideran agentes reactivos.

Un interactor es un tipo de agente que se comunica con el usuario directamente. Para ello, proporciona una representación perceptual de su estado interno y además un medio para modificarlo.

Un objeto es un concepto que hace referencia a una entidad computacional con un estado local. Sin entrar en la definición de éste, simplemente señalar la siguiente relación entre los tres términos señalados (Coutaz, et al., 1993a): Un interactor es un agente, y éste es un objeto.

3.3.4.2. Los modelos de agentes múltiples

Las arquitecturas conceptuales tratadas aquí coinciden en ver los sistemas interactivos como un conjunto de agentes cooperantes que intercambian información entre sí y con el usuario. Dentro de ellos, existe una distinción básica presente de una manera u otra consistente en diferenciar entre un núcleo funcional y un interfaz de usuario, lo cual garantizaría la separación de estas capas y la modificabilidad separada de ellas.

En concreto, las entidades consideradas por los modelos (Coutaz, et al., 1993a) atendiendo a la diferenciación anterior son las mostradas en la figura 3.15 :

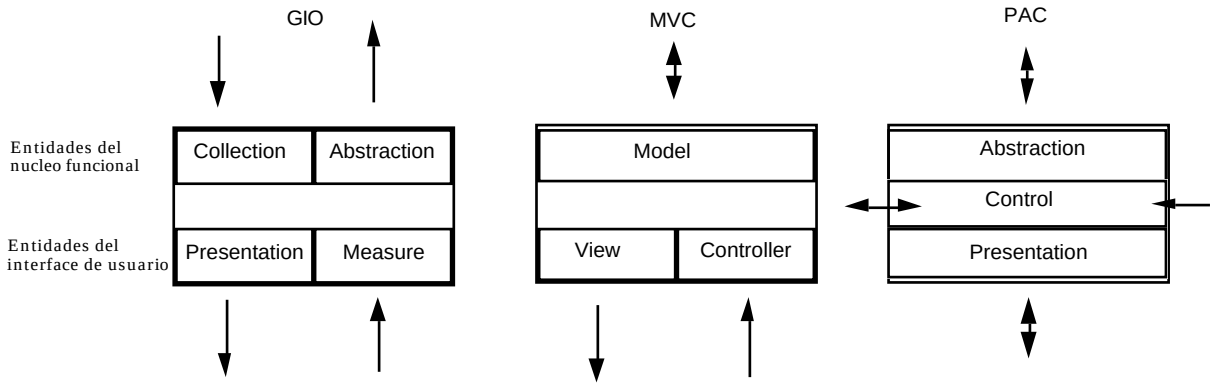


Figura 3.15: Una comparación tomada de (Coutaz, et al., 1993a) de los conceptos incluidos entres arquitecturas conceptuales multiagente. Los nombres de los términos no han sido traducidos para respetar los matices propios de cada uno de ellos.

En GIO (Graphical Interaction Object) (Faconti, 1993; Paterno, 1993; Paterno, 1994) un agente posee cuatro perspectivas: Collection and Abstraction interactuan con el núcleo funcional y Measure y Presentation cubren los aspectos de input y output con el usuario.

En MVC (Model, View y Controller), un agente tiene su competencia funcional definida en el modelo, y, por otro lado View y Controller definen el interface de usuario.

En PAC (Coutaz, 1987; Coutaz, et al., 1993b; Nigay, et al., 1993) los componentes de Abstraction y Presentation corresponden en un agente a las funciones que los otros modelos consideran, es decir, comunicación con los componentes funcionales y con el interfaz de usuario. Como elemento extra, esta arquitectura añade la capa de Control, encargada de poner en contacto a las otras dos así como de coordinarse con la conducta de otros agentes.

3.4. Modelización del usuario

Las descripciones que se basan en la modelización del usuario están basadas fundamentalmente en el análisis de las operaciones internas que los usuarios realizan a la hora de aprender a utilizar, o simplemente manejar, un interfaz. De este modo, aunque el dispositivo o interfaz es descrito de en cierto modo, el foco central de la notación se encuentra en el usuario.

Las propuestas de notaciones dentro de este campo pueden ser clasificadas en tres tipos. Gramáticas, las cuales intentan especificar los conjuntos de reglas que definirían el lenguaje de interacción con el dispositivo, intentando además encontrar la forma en que esas reglas fueran organizadas adecuadamente desde un punto de vista cognitivo; reglas de producción, que estructurarían las descripciones siguiendo reglas del tipo si X entonces Y; y, por último, modelado cognitivo de los errores que abarca notaciones dirigidas a analizar de una manera detallada aquellas situaciones en las que los usuarios encontrarían sistemáticamente problemas a la hora de manejar un interfaz.

3.4.1. Gramáticas

Una gramática es un conjunto de reglas que definen la forma en que pueden ser construidas las distintas sentencias de un lenguaje. En este caso, hablamos de un lenguaje de input, por lo que una gramática define de qué manera podemos construir comandos u ordenes correctamente de tal modo que el sistema informático sea capaz de aceptarlas.

Dentro del apartado de gramáticas hemos incluido varios modelos. El primero de ellos es importante porque constituye prácticamente la primera aportación al

campo de los métodos formales de descripción de la interacción hombre-ordenador (el lenguaje de acción de Reisner). No obstante, no ha sido prácticamente utilizado, habiendo recibido un número bastante grande de críticas. El segundo de los modelos (la gramática de múltiples partes de Shneiderman-Shneiderman, 1982-) es de hecho una modificación al método anterior, que, aunque corrige algunos aspectos de aquel, no ha sido utilizado en ninguna aplicación conocida.

El tercero de los métodos (la gramática de acción-tarea TAG) puede ser considerado como la aportación gramatical más madura al área. Un buen número de extensiones han sido propuestas posteriormente, las cuales son tratadas en el apartado posterior.

3.4.1.1. El lenguaje de Acción de Reisner

El lenguaje de acción de Reisner (Reisner, 1981) está dirigido a expresar las conexiones entre acciones y tareas por medio de una notación denominada Backus-Naur Form (BNF), la cual es una notación standard utilizada para describir qué expresiones de lenguajes de programación tales como Algol 60 o Pascal son correctas o no (de Haan, et al., 1991; Kirakowski y Corbett, 1990; Newell y Simon, 1972a). El trabajo de Reisner consiste en una adaptación de esta notación que permita modelar los "lenguajes de acción" de los sistemas interactivos (secuencias de botones, movimientos de tecleado), estando particularmente interesados en los factores cognitivos de los usuarios (lo que los sujetos tienen que aprender y recordar).

En breves palabras, su gramática se compone de los siguientes elementos:

1) Un conjunto de símbolos terminales (las palabras del lenguaje). Se escriben en mayúsculas. Un símbolo no terminal es una primitiva y no puede ser descompuesto en símbolos de nivel inferior.

2) Un conjunto de símbolos no terminales (constructos inventados para mostrar la estructura del lenguaje de comandos). Se escriben en minúsculas. Un símbolo no terminal es descomponible en símbolos terminales.

3) Un símbolo inicial. Este símbolo tareas de nivel superior que el usuario debe realizar. Este símbolo debe ser descompuesto hasta alcanzar las primitivas que finalmente indican los pasos concretos para llevar a cabo la tarea.

4) Los metasímbolos "+", "|" y ":=" (con el significado normalmente de "y", "o" y "compuesto de".

5) Reglas construidas a partir de lo anterior (p.e. dibujar-círculo := seleccionar-herramienta-círculo + POSICIONAR-CURSOR + APRETAR-BOTON + MOVER-RATON + SOLTAR-BOTON).

```
emplear_Dn                                  ::= (recuperar_info._sobre
                                            sintaxis_Dn) + utilizar_Dn

(recuperar_info._sobre_sintaxis_Dn)         ::= (recuperar_de_la_memoria
                                            humana)|(recuperar_de_fuente
                                            externa)

(recuperar_de_la_memoria_humana)
                                            ::=(RECUPERAR_DE_MEMORIA_LARGO_P
                                            LAZO)

                                            | (RECUPERAR_DE_MEMORIA_
                                            CORTO_PLAZO)

recuperar_fuente_externa                    ::=RECUPERAR_DE_LIBRO

                                            |PREGUNTAR_ALGUIEN

                                            |EXPERIMENTAR

                                            |USAR_AYUDA_ON-LINE

utilizar_Dn                                 ::=identificar_primera_linea

                                            +escribir_comando_Dn

                                            +APRETAR_ENTER

identificar_primera_línea                   ::=mover_ratón+APRETAR_BOTON

escribir_comando_Dn                         ::=TECLEAR_D+teclear_n
…
```

Figura 3.16: Descripción por medio del lenguaje de acción de Reisner de la operación Dn (borrar n lineas). Tomado de de Haan et. al (de Haan, et al., 1991).

Posteriormente a las reglas comentadas anteriormente, Reisner introdujo una forma para distinguir entre acciones cognitivas y acciones observables del usuario. Las primeras se escriben entre paréntesis y las segundas en mayúsculas Y cursivas. A continuación se muestra un ejemplo que implica el uso de un comando que borra cierto número de líneas (Dn: Delete n lineas).

En el ejemplo, el comando a utilizar (el símbolo inicial) necesita dos cosas: determinar la información sobre la sintaxis y luego utilizar esa información. Para recuperar la información existen diversas fuentes y una vez este paso es terminado se pasa a ejecutar la tarea (utilizar Dn ya no está en cursiva por lo que es una acción externa). Hay tres pasos de los cuales sólo uno es un símbolo terminal (APRETAR_ENTER). El resto necesita ser descompuesto hasta llegar a las primitivas finales. Esta parte final no ha sido terminada del todo ya que todavía sería necesario especificar los pasos para mover_ratón y para teclear_n.

Estas reglas permitirían describir el conjunto de acciones posibles que el usuario podría llevar a cabo, determinando en definitiva lo que el usuario tiene que saber (de Haan, et al., 1991). Por otro lado, Reisner sugiere tres medidas de complejidad del lenguaje de interfaz que pueden ser extraidas a partir de su lenguaje de especificación: 1) número de diferentes símbolos terminales, 2) longitud de las cadenas terminales para tareas dadas y 3) número de reglas necesarias para describir la estructura del lenguaje de interfaz.

Esta notación, no obstante, ha sido muy poco utilizada. Sólo su propia autora y un estudio posterior de Richards et. al. (Richards, et al., 1986) lo han hecho. En el primero de ellos, Reisner demuestra que un sistema informático es mejor que otro de funcionalidad similar y que se había propuesto mejorar. Un estudio empírico apoya sus afirmaciones.

Algunas de las críticas que se han dirigido contra este formalismo son las siguientes:

1) No hace referencia a la diferencia entre estado y transición tal y como ocurre en una notación de diagramas de estados. Además, es más difícil de visualizar que aquella (Jacob, 1986).

2) No contempla la concurrencia de acciones, no es apta para ser ejecutada, no permite relacionar el input con sus efectos y no es suficiente para describir el nivel sintáctico de los diálogos (Alexander, 1990; Alexander, 1987).

3) No hay mecanismo para describir sucesos del contexto.

4) No es posible describir los mecanismos de presentación en pantalla.

5) No hay manera de describir formalmente la funcionalidad del sistema, es decir las acciones y respuestas del sistema (Jacob, 1983).

6) La elección de símbolos no terminales es dejada a la elección del diseñador, por lo que el número de reglas es arbitrario (Kirakowski y Corbett, 1990).

7) No es posible describir sistemas en los que la interacción puede producirse de un modo no lineal, produciéndose solapamientos en la especificación de distintos comandos, tal y como ocurre en los interfaces de manipulación directa.

Otras críticas mencionadas en Curry et. al. (Curry y Monk, 1990b).

- No permite distinguir entre modos de responder y entre diálogos simultaneos-concurrentes. Scott y Yapp (1988) han propuesto modificaciones.
- No es posible describir diálogos de manipulación directa.
- Se limita la descripción de las acciones del usuario sin tener en cuenta las respuestas del sistema.
- Es bastante difícil creer en que de alguna manera realmente represente conducta cognitiva.
- El alcance de la especificación no es tan amplio como en CLG de (Moran, 1981b).
- Es difícil hacer cambios sobre una especificación.
- Es relativamente ininteligible sin práctica y además es difícil captar la dinámica de un interfaz.

En definitiva, según Richards et. al (1986) el lenguaje de Acción de Reisner sólo es capaz de describir interacciones que sean estrictamente seriales y en las que las tareas sean realizadas una por una, y la tarea actual no sea afectada por ninguna tarea previamente realizada.

Una extensión realizada a la gramática de Reisner es la Gramática de múltiples partes de Shneiderman. (Shneiderman, 1982) el cual presenta las siguientes modificaciones. Estas son:

1) Incluir las respuestas del sistema a las acciones de los usuarios. Esto se llevaría a cabo etiquetando las reglas en función de qué parte estuviera actuando en qué momento. Esto daría lugar a una gramática con dos partes, pero deja

abierta la posibilidad de introducir más etiquetas para modelar sistemas con más partes (p. e. groupware).

2) Definir cómo cambiar las características visuales de la pantalla.

3) Permitir declaraciones de ventanas en la pantalla y operaciones dinámicas tales como movimientos del cursor, etc.

La propuesta de Shneiderman no ha sido posteriormente elaborada en ningún lugar (de Haan, et al., 1991). Jacob (1983) opina que un diagrama de transición de estados podría modelar perfectamente las mismas situaciones que esta gramática, teniendo además otras ventajas que ésta no tiene. En general, prácticamente todas las críticas que podrían hacerse al lenguaje de Reisner salvo la 5, y en parte la 4 son todavía aplicables a esta versión modificada.

3.4.1.2. TAG de Green y Payne

TAG (abreviatura para Task-Action Grammar) de Green y Payne es sin duda la aportación más relevante hecha desde la perspectiva gramatical al campo de la formalización del interfaz del usuario desde el punto de vista psicológico. Así, desde una propuesta inicial, sus autores han ido añadiendo mayor cantidad de pruebas empíricas, extensiones y descripciones de aplicaciones de la vida real hasta conformar una metodología de gran interés y alcance.

Trataremos en esta exposición los siguientes aspectos: en primer lugar, introduciremos el concepto de consistencia el cual es central en su modelo y la notación dirigida a describirla, en segundo lugar, las bases teóricas y empíricas esgrimidas por los autores así como estudios de otras fuentes, en tercero, examinaremos los aspectos no analizados y las extensiones que los propios autores han sugerido, y, finalmente, la compararemos con otras notaciones existentes.

3.4.1.2.1. La noción de consistencia y la notación para describirla.

La noción de consistencia.

El principio de consistencia en el interfaz es considerado como central para el diseño de sistemas informáticos. Por ejemplo, la mayoría de las colecciones de normas de buen diseño suelen incluir este principio (Apple, 1987; Smith y Mosier, 1986).

No obstante, para Green y Payne, la definición de este concepto ha sido muy ambigua, dejando su interpretación y aplicación concreta a merced de la apreciación subjetiva de cada diseñador (Schiele y Green, 1990). Así, aunque es sabido que la noción de consistencia hace referencia de algún modo a regularidades percibibles en los interfaces de las aplicaciones, tales como las acciones que el usuario tiene que ejecutar para llevar a cabo sus tareas, el feedback que proporciona el sistema, la disposición espacial de la pantalla, etc., no es sencillo operacionalizar esas intuiciones en términos observables y propiedades medibles.

El punto importante (Schiele y Green, 1990) es que los usuarios sean capaces de percibir y utilizar en su beneficio las regularidades percibidas en el interfaz, ya que esto les permite llevar a cabo generalizaciones del conocimiento parcial del sistema a otras áreas. Es decir, el usuario pueda reutilizar conocimientos adquiridos en otras partes del sistema a las nuevas situaciones, dado que las reglas del interface y su lógica subyacente siguen siendo la misma. Esta definición lleva a considerar la consistencia como producida por medio de la representación mental que el usuario tenga del sistema, es decir, su conocimiento del dispositivo. Por ello, una afirmación central en TAG es que la consistencia sintáctica en los lenguajes de comandos sólo será aprovechada si se la hace coincidir con las categorías semánticas de las tareas.

Un repaso a la clasificación de los distintos tipos de consistencia según Green y Payne (Payne y Green, 1986) nos permitirá ampliar su enfoque. Esta taxonomía se hace desde el punto de vista del nivel de descripción lingüística en el cual aparecen ciertas características de interés. No obstante, ellos advierten que una clasificación realmente útil sería aquella establecida en términos de teoría psicológica, no lingüística.

1) Consistencia sintáctica. La consistencia sintáctica alude al caso en que, por ejemplo, todos los comandos son ejecutados a través de una misma estructura como puede ser una letra más un terminador. Este tipo de consistencia fue el objetivo de la gramática de conjuntos, una variante de la gramática de dos niveles de van Wingaarden (Green y Payne, 1984), la cual consiguió predecir el comportamiento de unos sujetos en una serie de lenguajes artificiales mínimos. No obstante, esta gramática presentaba el problema de no tener ningún mecanismo que relacionara los aspectos sintácticos con las propiedades semánticas, estando pues limitada su utilidad como modelo cognitivo. TAG (que será presentada posteriormente) supone un intento por superar este problema.

2) Consistencia léxica. En lenguaje natural, los lexemas están ligados a su significado por medio de conexiones arbitraras. En los lenguajes de comandos (Carroll, 1982) se ha demostrado que un principio beneficioso es el de congruencia (el ajuste entre características léxicas y el significado de asignado). Por ejemplo, un icono con una determinada forma debería ajustar con una función congruente a su forma.

3) Alineamiento sintáctico-semántico. En un lenguaje consistente ideal, el alineamiento no debería producirse únicamente entre cada término y su significado por separado, sino que debería darse en su estructura como un conjunto. Así, Green y Payne (1984) mostraron que en un lenguaje de comandos comercial la existencia de principios organizacionales distintos en el uso de las teclas modificadoras llevaba a los usuarios a tener problemas que fueron solucionados utilizando un único principio organizador para todo el lenguaje.

4) Consistencia semántica. La noción de consistencia semántica puede igualarse a la de completud y significa lo siguiente: un lenguaje de comandos proporciona una serie de expectativas acerca de sus posibilidades, las cuales debería ser satisfechas por el sistema. Por ejemplo, si un procesador de textos permite la búsqueda de palabras hacia arriba y hacia abajo, su función de reemplazar texto debería funcionar del mismo modo.

La notación

TAG es una notación que ofrece un método para captar la consistencia. Los elementos principales de TAG en su formulación más reciente (Schiele y Green, 1990) son los siguientes:

1) Una lista de dimensiones y sus posibles valores. Estas dimensiones distinguen a las operaciones entre sí. Por ejemplo, borrar en el sistema operativo MS-DOS un directorio o un archivo implica distintos valores (archivo/directorio) que difieren a lo largo de la dimensión entidad_borrada.

2) Un diccionario de tareas simples definidas en función de las dimensiones y los atributos anteriormente definidos.

3) Las primitivas o símbolos terminales que expresan acciones del usuario. Estas son las acciones a nivel más básico.

4) Unas reglas-tareas que prescriben como las tareas simples se expanden en acciones primitivas y/o en reglas-subtareas (las cuales son también especificadas hasta terminar teniendo únicamente acciones primitivas).

Mostraremos como ejemplo parte de la descripción del procesador de textos MacWrite (Schiele y Green, 1990). Esta parte hace referencia a las operaciones de copiar, cortar y borrar textos y palabras

1) Lista de dimensiones que permiten diferenciar a estas operaciones entre sí y valores que pueden tomar. Las tareas vemos que, por ejemplo, pueden diferir en las unidades a las que hacen referencia (un carácter o una palabra) o en el efecto que producen (borrar, reemplazar, etc.).

Dimensiones	Valores
Unidad	Carácter/palabra
Extensión	Número de unidades involucradas en la acción
Efecto	Añadir/Borrar/Reemplazar/Copiar
Usa clipboard4	Si/No

2) Tareas simples definidas en función de las dimensiones. Las tareas simples que vamos a describir son las siguientes:

(1) Borrar texto.
Unidad=caracter, extensión=cualquiera, Efecto=Borrar, Clipboard=No.
(2) Borrar palabra.
Unidad=palabra, extensión=1, Efecto=Borrar, Clipboard=No.
(3) Cortar texto (borrar palabra llevándola simultaneamente a un buffer especial denominado clipboard).
Unidad=caracter, extensión=cualquiera, Efecto=Borrar, Clipboard=Si.
(4) Copiar texto (copiar texto en un buffer especial denominado clipboard sin borrarla).

---

4 El clipboard es un area especial de memoria que almacena objetos recien borrados para reutilizarlos inmediatamente en caso de ser necesario.

Unidad=caracter, extensión=cualquiera, Efecto=Copiar, Clipboard=Si.

(5) Reemplazar texto.

Unidad=caracter, extensión=cualquiera, Efecto=Reemplazar, Clipboard=No.

(6) Pegar texto sobre texto anterior (llevando el texto desde el buffer denominado clipboard).

Unidad=caracter, extensión=cualquiera, Efecto=Reemplazar, Clipboard=Si.

Vemos que, por ejemplo, borrar un texto afecta a un número indeterminado de caracteres(Unidad=caracter, extensión=cualquiera), tiene el efecto de borrar y no implica Clipboard, por lo que el texto no podrá ser pegado en otro sitio distinto. Cortar texto es igual a borrar palabra sólo que el texto va al clipboard. Las otras funciones son definidas similar mente.

3) Las primitivas serían las siguientes:

- Señalar (dirigir la flecha del ratón al lugar apropiado).
- Arrastrar (presionar el botón del ratón, dirigir la flecha del ratón al lugar apropiado sin dejar de presionar, soltar el botón del ratón).
- "Clickar" (presionar el botón del ratón e inmediatamente liberarlo).
- Doble "clickar" ("clickar" dos veces).
- Teclear (%caracteres) (escribir un número dado de caracteres por medio del teclado).
- Menú (utilizar un comando de un menú).
- "tecla" (presionar una tecla dada).

4) A partir de esos elementos, podemos extraer la regla-tarea que subyace a todas esas tareas en el procesador de textos MacWrite:

Tarea[Unidad= carácter/ palabra, Extensión= cualquiera, Efecto= Borrar/ Copiar/ Reemplazar, Clipboard= cualquiera] := seleccionar [Unidad= carácter/ palabra, Extensión= cualquiera] + editar [Efecto= Borrar/ Copiar/ Reemplazar, Clipboard= cualquiera]

Este es el punto central de la notación. Como vemos, las tareas señaladas pueden considerarse como altamente consistentes, ya que una sola regla es capaz de describir completamente la estructura en la que es posible enviar inputs al sistema. Esta estructura corresponde a la subtarea seleccionar más la tarea editar la cual subyace a una variedad bastante amplia de comandos diferentes.

Esta regla, al ser tan simple, sería supuestamente fácil de entender y recordar por los usuarios. Otros sistemas más inconsistentes no podrían resumirse de una manera tan simple.

Dependiendo de las características concretas que correspondan a las dimensiones consideradas estas subtareas corresponderán a unas u otras acciones primitivas

En concreto, esta regla-tarea alude a las siguientes subreglas-tarea:

```
seleccionar [Unidad= caracter, Extensión= cualquiera] := señalar + arrastrar.
seleccionar [Unidad= palabra, Extensión= 1] := señalar + doble"clickar".
editar [Efecto= Borrar, Clipboard= no] := "Tecla borrado".
editar [Efecto= Borrar, Clipboard= si] := Menu (edición-copiar) | "Comando-V"
editar [Efecto= Copiar, Clipboard= si] := Menu (edición-copiar) | "Comando-C"
```

Así vemos que, por ejemplo, cuando se trata de seleccionar un carácter con una extensión cualquiera (un texto) es necesario señalar más arrastrar (por ejemplo, para borrar texto o para cortar texto). Cuando se tratara de seleccionar una palabra sería necesario seleccionar la palabra y doble"clickar".

De este modo, el lenguaje de interacción de MacWrite podría ser reducido en su totalidad a una serie de reglas generales de este tipo. El lector interesado puede consultar Schiele et. al (1990) para ver el ejemplo completo, así como otros ejemplos relacionados.

A continuación mostramos un ejemplo que sirve para revelar una inconsistencia en la forma de especificar la copia de archivos en el sistema operativo de los ordenadores Macintosh.

Para realizar una copia de un archivo o mover un archivo, las dimensiones relevantes y sus atributos serían:

Dimensiones	Características
Dejarcopia	Si/No
Disco	Mismo/distinto

Las tareas simples serían:

(1) Copiar archivo otro disco

Dejarcopia=si, disco=distinto

(2) Mover archivo mismo disco

Dejarcopia=no, disco=mismo

(3) Copiar archivo mismo disco

Dejarcopia=si, disco=mismo

(4) Mover archivo otro disco

Dejarcopia=no, disco=distinto

Entonces, las tareas generales serían:

Tarea[Dejarcopia=si, disco=distinto] := seleccionar [Dejarcopia=si, disco=distinto] + arrastrar

Tarea[Dejarcopia=no, disco=mismo] := seleccionar [Dejarcopia=no, disco=mismo]+ arrastrar

Tarea[Dejarcopia=no, disco=distinto] := seleccionar [Dejarcopia =no, disco= distinto] + arrastrar + seleccionar[Dejarcopia =no, disco= distinto] +arrastrar + (menuedición-vaciar papelera) + señalar [Dejarcopia=no, disco=distinto] + (presionarbotón OK)

Tarea [Dejarcopia=si, disco=mismo] :=seleccionar [Dejarcopia =si, disco=mismo] + arrastrar

Y las subtareas son:

Seleccionar[Dejarcopia=si,disco=mismo]:=señalar+clickar+"teclaopción"

Seleccionar [Dejarcopia=si, disco=distinto] := señalar+clickar

Seleccionar [Dejarcopia=no, disco=mismo]:= señalar+clickar

Seleccionar [Dejarcopia=no, disco=distinto]:= señalar+clickar

Como vemos, cuatro tareas simples han dado lugar a cuatro tareas generales ya que no existe una manera sencilla de colapsar las distintas subreglas en una sola. La dificultad se encuentra en la acción de mover un archivo de una unidad de disco a otra sin dejar una copia detrás. Esta acción resulta muy compleja ya que implica realizar un borrado del fichero operación que es especialmente complicada para proteger al usuario de errores involuntarios. Una segunda fuente de dificultad estriba en que la selección del archivo cuando la operación es realizar una copia sobre el propio disco necesita que se apriete la tecla opción.

Medidas del interfaz

Green y Payne (Payne y Green, 1986; Schiele y Green, 1990) afirman que el índice de complejidad más importante derivado de una descripción basada en TAG es el número de reglas-tarea, ya que estas reglas definen la configuración de nivel superior del lenguaje que se analiza. Para dos lenguajes que poseyeran el mismo número de reglas-tarea, entonces deberían contarse todas las reglas, incluidas las subreglas-tarea. Este índice daría una medida de la facilidad de aprender un determinado interface, aunque, no en términos absolutos, sino sólo relativos, prediciendo únicamente qué diseño sería mejor de un conjunto de alternativas.

De hecho, existen otras limitaciones dignas de reseñar. En primer lugar, los propios autores dudan de la validez de las especificaciones en TAG para analizar diseños completos debido a que cuanto más grandes y distintas sean las cosas a comparar más difíciles son las comparaciones. Por ello, afirman que su interés recae ante todo en el análisis de partes pequeñas del sistema en las que la funcionalidad es prácticamente equivalente y es posible elegir los detalles concretos del lenguaje de interfaz. En segundo lugar, no existen hoy por hoy medios que automaticen la tarea de llevar a cabo análisis TAG, por lo que, aunque fueran de utilidad, llevar a cabo descripciones de sistemas complejos sería una tarea excesivamente ardua.

Otras cosas que TAG no hace son (Schiele y Green, 1990): 1) No describe los mecanismos cognitivos o motores por medio de los cuales son llevadas a cabo las tareas. Por ello, TAG no hace predicciones cuantitativas acerca del tiempo para aprender a manejar un interfaz. 2) No describe el proceso de planear secuencias complejas de acción. Las tareas simples de TAG no requieren esfuerzo de planeamiento. 3) No describe el lado de interpretación del ciclo de la acción. y 4) No describe suficientemente las entidades conceptuales en la mente del usuario5.

Tareas simples.

Una tarea simple es una tarea que el usuario puede ejecutar rutinariamente sin requerir una estructura de control separada, tal como la ramificación o la

---

5 Aunque trabajos posteriores de ese equipo investigador están ampliando ese punto.

iteración, que requiera monitorización del progreso del plan de acción. En este sentido, las tareas simples pueden ser consideradas como operadores dentro de la visión clásica de la solución de problemas (Newell y Simon, 1972a). No obstante, existe una diferencia importante, mientras los operadores mentales son conceptos atómicos, no relacionados entre sí salvo en cuanto a su papel dentro del camino hacia una solución, TAG trata las tareas simples como conceptos que pueden ser agrupados en categorías mentales (las reglas-tarea).

Los conceptos, tal y como los trata TAG, se representan por medio de conjuntos de atributos definitorios, siguiendo la visión clásica, a pesar de la existencia en la literatura de otras alternativas, sin intentar por ello dar por terminado el debate existente acerca del tema.

Un atributo es definido como algo que puede tomar un valor con respecto a un término. Estos atributos, y sus posibles componentes, deberían tener validez psicológica en tanto en cuanto sean importantes para la categorización del mundo. No obstante, no hay nada que el analista pueda hacer para asegurar este ideal salvo apoyarse en su intuición. Por otro lado, los posibles valores que los atributos pueden tomar deberían ser especificados uno por uno, siendo su rango de valores normalmente pequeño (Payne y Green, 1986).

3.4.1.2.2. Justificación empírica.

Green y Payne (Green y Payne, 1984) aportaron la primera de las pruebas de su modelo mediante un estudio de aprendizaje de pares asociados de cuatro lenguajes de comandos. Tres de ellos eran lenguajes artificiales, diseñados del siguiente modo: El número 1 presentaba etiquetas mnemotécnicas en los comandos y además estructura-consistencia, el 2 sólo consistencia-estructura y el 3 sólo comandos mnemotécnicos. El cuarto lenguaje era un subconjunto de un lenguaje comercial el cual presentaba códigos mnemotécnicos pero tenía dos principios conflictivos para organizar su estructura. Los resultados, en línea con lo predicho por los autores, confirmaron que el lenguaje número 4 era claramente el peor de todos, mientras que el 1 no era mejor que el 2 en una de las medidas (recuerdo libre) pero sí que el 3, y mejor que ambos en otra de las medidas (recuerdo inducido).

El estudio anterior fue replicado con posterioridad (Cramer, 1990) corrigiendo algunos aspectos. En primer lugar, se añadió un cuarto lenguaje artificial (ni

estructura ni códigos mnemotécnicos) de modo que se completó un diseño factorial 2 X 2. En segundo lugar, se modificó el lenguaje 2 para controlar la estrategia de generar códigos mnemotécnicos supuestamente utilizadas por los sujetos en el experimento original. En tercer lugar, se añadió una tarea de categorización.

Los resultados no soportaron los supuestos de TAG. El factor de nemotecnia sí que presentó diferencias significativas a favor de ésta, mientras que el factor de estructura-consistencia no. La única evidencia a favor de TAG provino del estudio de categorización, el cual mostró que en las condiciones con estructura los sujetos tendían a agrupar los comandos siguiendo la organización del lenguaje, mientras que en los otros no. Esto hace suponer que las nociones de abstracción y generalización de TAG juegan algún papel, y que los sujetos explotan estas claves de algún modo. Por otro lado, sus resultados vinieron a realzar el papel de la nemotecnia como un factor de importancia en la construcción de lenguajes de comandos.

En otro estudio (Payne, 1988) comprobó dos métodos para abreviar comandos. En uno de ellos, se ofreció una regla general que abarcaba el mayor número de casos posibles y aquellos imposibles de cubrir se trataron como excepciones que debían aprenderse individualmente. En el otro, se dividió el lenguaje según categorías semánticas y se ofrecieron dos reglas distintas, una para cada categoría, de modo que no aparecieron excepciones. Sus resultados apoyaron este último método de abreviación, y por tanto la lógica subyacente a TAG.

Payne y Green (1989) desarrollaron tres lenguajes dirigidos a probar una afirmación central de TAG, que la consistencia sintáctica no produciría mejores resultados si no se encontraba alineada con las categorías semánticas de la tarea. Así, desarrollaron tres lenguajes artificiales para las siguientes tres tareas llevadas a cabo por un mismo sistema informático: solicitar información, enviar mensajes y jugar a juegos. El primero de los lenguajes (1) presentaba una única regla para generar comandos a través de las tres categorías, el segundo lenguaje(2) utilizaba tres reglas, las cuales coincidían con cada una de las tareas, el tercer lenguaje (3) utilizaba sólo dos reglas, pero su utilización no era consistente dentro de las tareas. Sus resultados confirmaron las predicciones de TAG, produciendo una ordenación en los lenguajes de este tipo 1 de mejor a peor de este tipo 1, 2 y 3. De particular importancia es el hecho de que el lenguaje 2 fuera mejor que el 3, ya que, aún teniendo un mayor número de reglas, el hecho de que su aplicación

coincidiera con cada una de las tareas, llevó a una ventaja sobre un lenguaje con menos reglas pero aplicadas independientemente de las categorías semánticas de la tarea.

3.4.1.2.3. Tratando problemas de tamaño real.

Schiele y Green (1990) se plantearon el análisis de un problema de tamaño real. Para ello analizaron tres programas del entorno Macintosh y luego generaron una serie de reglas-tarea que abarcarían esas aplicaciones. El resultado podría considerarse un éxito ya que a través de este análisis es posible revelar parte del fundamento que sustenta el "estilo" de este sistema operativo. Además, es posible apreciar una serie de pequeñas inconsistencias entre las aplicaciones que de otro modo pasarían inadvertidas.

No obstante, este análisis obligó a los autores a introducir una serie de elementos no presentes anteriormente en la notación tales como restricciones en la co-ocurrencia de atributos, el uso de variables importadas, el reasignamiento temporal (que supondría un mecanismo parecido al de una subrutina) y la iteración de componentes dentro de la tarea. Estos elementos según los autores disminuyen un tanto la credibilidad psicológica de los modelos TAG debido a que postulan mecanismos que no está probado existan en los seres humanos. No obstante, si no fueran introducidos estos modelos en su conjunto perderían plausibilidad al generar excesivas reglas.

3.4.1.3. Extensiones a TAG

3.4.1.3.1. D-TAG de Howes y Payne.

TAG no tenía ningún mecanismo para determinar el estado de la pantalla a la hora de realizar una descripción. D-TAG es una notación que intenta ampliar este aspecto de TAG. En la opinión de Howes y Payne (Howes y Payne, 1990) tampoco GOMS en la versión de Kieras y Polson (que se verá más adelante) es capaz de modelar adecuadamente el entrelazado entre el estado de la pantalla y las acciones de los usuarios que caracteriza los interfaces actuales.

Una investigación que Howes y Payne consideran clave con respecto a este punto es la de Mayes et. al (Mayes, et al., 1988). En ella, se mostró que usuarios expertos de un editor de textos orientado a la pantalla (MacWrite) no eran capaces

de recordar los nombres de comandos que utilizaban normalmente. Esto mostraba que su habilidad estaba basada en el reconocimiento: los usuarios aprenden a seleccionar e interpretar la información sobre la pantalla cuando van a realizar sus tareas en lugar de tener una serie de reglas en mente a partir de las cuales van derivando los comandos concretos apropiados para cada ocasión.

Howes y Payne (1990) introducen dos modificaciones a TAG para permitirle captar estas consideraciones. Estas son:

• El operador item_pantalla ((atributos_tarea) (atributos_pantalla)). Este operador toma el subconjunto de objetos en la pantalla, analiza sus características semánticas y las compara con las características de la tarea que está llevándose a cabo, devolviendo el resultado que parezca más apropiado.

Un problema de este operador es cómo describir las características de la pantalla. Para ello, Howes y Payne postulan una serie de atributos que servirían para describir ítems en la pantalla. Estos serían: la clase de objeto (texto, objeto geométrico, etc.), su localización (localización horizontal=derecha, vertical=arriba, etc.; es posible utilizar más de un localizador), estado (también son posibles varios valores) y su forma.

• El operador efecto ((objeto,) (asignamiento_atributo)) permite considerar los efectos laterales de una acción sobre la pantalla permitiendo detectar inconsistencias en el resultado que se produce en la pantalla. Un ejemplo es un programa de gráficos orientados al objeto (MacDraw) en el que las herramientas de dibujo pueden ser modelados consistentemente salvo por el hecho de que la herramienta de texto permanece seleccionada después de haber sido utilizada a diferencia de todas las otras que son deseleccionadas. El operador efecto permite representar este efecto lateral de tal modo que se puede mostrar la consistencia/inconsistencia de este comportamiento.

3.4.1.3.2. El modelo de aprendizaje de menús Ayn de Howes.

Howes (Howes, 1994) intenta lograr un modelo que dé cuenta de tres aspectos muy generales de la conducta humana en relación al aprendizaje de estructuras de menús. En primer lugar, la gente aprende los dispositivos por exploración. En segundo, la gente mejora con la práctica. Y en tercero, la gente adquiere conocimiento basado en la pantalla en el sentido discutido en el apartado dedicado a D-TAG.

Su modelo utiliza la arquitectura cognitiva denominada SOAR (Laird, Rosenbloom y Newell, 1987) que respaldaría sus aportaciones. Estas son fundamentalmente un procedimiento de decisión entre acciones posibles basado en el conocimiento extraído de cuatro fuentes. Esas fuentes son:

1) Conocimiento acerca de la proximidad semántica: Como primer paso para realizar un análisis se le proporciona a Ayn una lista de ítems de menús que el analista cree que tienen alguna relevancia semántica con respecto al sistema. Estos no son los ítems correctos, sino todos aquellos ítems que el novato consideraría semánticamente similares. Por ejemplo, si desearemos formatear un documento a dos columnas utilizando el procesador de textos Word 4.0 los siguientes ítems serían relevantes: Archivo, Ajustar Página, Ver, Formato, Sección, Documento, Columna 2 (el correcto), Ventana y Párrafo.

2) Conocimiento de detección de fallos: Cuando Ayn ha intentado todas las posibilidades con cierto ajuste semántico y no ha obtenido ningún resultado positivo Ayn intenta realizar la acción Cancelar. Si esta no es posible busca acciones alternativas como son Fuera (para liberar un menú) o Reiniciar.

3) Conocimiento de reconocimiento. Este es conocimiento que indica si algo ha sido visto anteriormente. Durante la exploración Ayn adquiere una base de datos de reglas de reconocimiento que ayudan a determinar si alguna parte ha sido ya explorada.

4) Conocimiento de control: Este conocimiento indica si es positivo o negativo aplicar un operador determinado. Este conocimiento es más difícil de adquirir ya que la información necesaria para decidir si una acción ayuda a alcanzar el objetivo deseado, a menudo está bastante alejada del punto en el que actualmente estamos trabajando. La estrategia de aprendizaje que según Howes mejor se adapta a esta situación es "final-first learning". Esta estrategia implica que cuando una acción alcanza el objetivo deseado se produce una conexión entre ambos. Más tarde, cuando se intenta alcanzar ese objetivo de nuevo, el modelo reconoce la acción que alcanzó el objetivo y considera como acertada aquella acción que a su vez alcanza a aquella. Esta serie de encadenamientos termina por establecer un método que puede ser ejecutado de modo enteramente automático.

Las cuatro fuentes de conocimiento son usadas en un procedimiento de decisión que tiene la forma de una serie de reglas de producción. Estas reglas establecen la jerarquía de aplicación de las distintas fuentes. En resumen, el

proceso consiste en primero intentar aplicar conocimiento de control positivo, si este no existe, entonces se comprueba sin en el pasado el objetivo fue alcanzado. Si es así, entonces la regla consiste en utilizar sólo acciones familiares y desechar las no familiares. En otro caso, se intentarán aquellas acciones que demuestren un ajuste semántico con el objetivo deseado. Por último, cuando ninguna otra de las acciones es aceptable, se aplican las acciones destinadas a cancelar o volver atrás.

Un modelo construido siguiendo estas indicaciones muestra, sobre sucesivos intentos, diversas conductas que finalizan con el objetivo deseado. A través de estos intentos se va produciendo una reducción en el número de pasos que se van intentando (y por tanto en el tiempo para alcanzar el objetivo) que reproducen bien la conducta esperada de un usuario. Asimismo, el comportamiento del modelo a través de los diversos intentos es lo suficientemente variable como para que todo tipo de interesantes conductas aparezcan, dando soporte a la variabilidad individual. Por otro lado, distintos tipos de usuarios según su nivel de experiencia (novato, intermedio, experto) pueden ser modelados para examinarse su comportamiento peculiar.

3.4.1.3.3. APT (Agent Partitioning Theory) de Reisner

En fechas recientes Reisner (Reisner, 1990; Reisner, 1993) ha propuesto una notación que recoge las aportaciones de Payne y Green pero añadiéndole una serie de reflexiones acerca del concepto de consistencia. En su propuesta, no existe una definición simple que permita colapsar de una manera uniforme, en todas las situaciones y para todos los usuarios, un grupo de reglas-tarea en una regla mayor que las incluya. Este comentario puede ponerse en relación con la inherente fragilidad señalada por Green y Payne en las descripciones TAG. Así, aunque ellos sugieren utilizar una serie de normas de sentido común acerca de características cuya generalización podría ser aceptable, de ninguna manera logran zanjar la cuestión. Reisner acentúa ese problema señalando que las inconsistencias nunca podrán ser detectadas simplemente utilizando una descripción formal que las revele de una manera automática. Diversas descripciones pueden ser válidas para el mismo lenguaje de tarea.

El papel del diseñador pues será no sólo diseñar el lenguaje siguiendo una serie de principios que lo hagan consistente, sino también anticipar otras formas

alternativas de entender las regularidades y actuar en consecuencia, bien poniéndoles freno o bien aprovechándolas para sus intereses.

APT es una notación y una teoría para soportar estos análisis. Su interés fundamental radica en llevar a cabo explícitamente diversas particiones de los lenguajes de input, pero debido a que se asemeja en gran medida al propio TAG no hemos considerado importante incluir un ejemplo.

3.4.2. Sistemas de producción

3.4.2.1. Introducción

Los sistemas de producción son un conocido formalismo utilizado en Inteligencia Artificial y Psicología Cognitiva (Newell y Simon, 1972a; Simon, 1981). No obstante, en el campo de la HCI el interés en los sistemas de producción no se centra tanto en su valor como método de representación del conocimiento sino en utilizarlo como una notación para el diseño de sistemas interactivos. Los sistemas de producción ofrecen un medio de representación flexible para el componente de diálogo de un interfaz, aunque no tanto en cuanto a la presentación (Kirakowski y Corbett, 1990).

Un sistema de producción se compone de los siguientes elementos:

a) Una base de reglas de la forma SI <condición> ENTONCES <acción>.

b) Una estructura de datos llamada contexto o memoria de trabajo.

c) Un interpretador que controla la actividad del sistema.

El funcionamiento de este sistema consiste en lo siguiente: en un primer ciclo, el interpretador evalúa cada una de las condiciones presentes en la base de reglas frente al contexto o memoria de trabajo; si encuentra una condición verdadera entonces la acción a la que se encuentra conectada es disparada y el control retorna al principio de la base de reglas para empezar una nueva fase de evaluación. En ciertas variantes, no obstante, el interpretador continuará hasta el final de la lista y en caso de hallar varias condiciones correctas tendrá que utilizar algún criterio para decidir cuál de ellas es disparada.

Las acciones pueden ser externas al sistema, pero también pueden ser internas, añadiendo datos a la memoria de trabajo, o nuevas reglas antes no presentes.

Veremos a continuación una serie de aproximaciones a la modelización de la interacción que aprovechan este esquema.

3.4.2.2. GOMS de Card, Moran y Newell

Este modelo ha sido probablemente el más influyente en el campo de la interacción hombre-ordenador, y, tras el paso de los años y ciertas modificaciones y añadidos, quizás sigue siendo el único que ha empezado a convertirse en una herramienta aplicada con grandes beneficios prácticos.

Este trabajo forma parte de las investigaciones realizadas en el centro de Palo Alto subvencionadas por Xerox y una prueba de la gran influencia que ese instituto jugó en el desarrollo de esta disciplina desde su creación en 1971 (Card, et al., 1983).

Antes de empezar con su descripción, señalar que haber incluido este modelo dentro del apartado de reglas de producción es una decisión un tanto inusual. Aunque los orígenes del modelo puedan rastrearse hasta los trabajos de Newell y Simon (Newell y Simon, 1972a), los cuales hicieron gran uso de esta metodología de representación, lo cierto es que GOMS no hace uso explícito de una arquitectura basada en reglas de producción. No obstante, extensiones posteriores sí han hecho uso explícito y, en definitiva, su estructura se acomoda perfectamente a este tipo de sistemas de descripción.

El plan de este apartado es el siguiente: En primer lugar veremos una introducción a GOMS y KLM, un modelo muy relacionado con aquel. En segundo lugar comentaremos algunos de los trabajos realizados posteriormente utilizando esta notación como marco general y por último veremos algunas de las críticas surgidas contra este modelo.

3.4.2.2.1. Descripción de las notaciones.

Para (Newell y Simon, 1972a) la estructura cognitiva de un usuario a la hora de realizar una tarea con un sistema interactivo consiste de cuatro componentes:

1) Un conjunto de Objetivos (**G**oals). Los objetivos son estructuras simbólicas que definen el estado de los asuntos que quieren ser logrados y determinan el conjunto de métodos posibles por el cual pueden llegar a ser obtenidos.

2) Un conjunto de Operadores (**O**perators). Los operadores son actos cognitivos, motores o perceptuales cuya ejecución es necesaria para cambiar cualquier aspecto del estado mental del usuario o afectar el contexto de trabajo. La conducta del usuario podría ser trazada como una secuencia de estas operaciones.

Los operadores son los que definen el grano de análisis al que el modelo GOMS ha sido realizado. En general, los operadores incorporan una cierta mezcla de mecanismos psicológicos básicos y conducta organizada aprendida, dependiendo la proporción del nivel al que el modelo haya sido realizado.

3) Un conjunto de Métodos para lograr los objetivos (**M**ethods).

Un método describe un procedimiento para alcanzar un objetivo. Es una de las maneras en las que un usuario almacena su conocimiento de una tarea y en el contexto del modelo GOMS se espera que este conocimiento haya sido adquirido previamente a la predicción que se va a intentar. No hay, o se espera que no haya, necesidad de crear planes durante la ejecución de la tarea, de tal modo que el componente de solución de problemas no sea una parte importante de la actividad del usuario.

4) Un conjunto de Reglas de Selección (**S**election rules).

Para alcanzar un objetivo, puede haber más de un método correcto para realizar la tarea. Por ello, usualmente es necesario un proceso que usualmente no es excesivamente largo ni complejo para decidir qué método es el más adecuado para cada situación. Por ejemplo, un usuario puede poder optar entre utilizar un ratón o utilizar las flechas del teclado para desplazarse a lo largo de un texto. Dependiendo de la distancia del desplazamiento, y eso el usuario indudablemente lo aprenderá con la experiencia, un método u otro puede ser más aplicable.

Los nombres de los cuatro componentes dan lugar al acrónimo GOMS (Goals Operations Methods and rules of Selection).

Estas cuatro entidades permiten realizar una descripción jerárquica de la forma en que un usuario realizaría una acción. En esta descripción, el usuario establecería un objetivo a realizar y extraería los métodos apropiados para llevarlo

a cabo. Normalmente, su objetivo no estará al nivel de comandos individuales, ya que si el usuario desea, por ejemplo, escribir su tesis, no existirán comandos en el ordenador que, de una manera directa, produzcan el resultado deseado[6]. Por ello, deberá desglosar su tarea en objetivos de nivel más reducido que acaben especificando operadores básicos (con un nivel igual al deseado para ese análisis particular).

Como una muestra del tipo de descripciones realizadas mediante GOMS véase la figura 3.17:

```
GOAL: Editar manuscrito.
 GOAL: Editar tarea unidad hasta que no haya más tareas
 unidad.
      GOAL: Adquirir-Tarea-Unidad
           Pasar a la siguiente página si se está al final
           de la página.
           Obtener la siguiente Tarea.
      GOAL: Ejecutar la siguiente tarea.
           GOAL: Localizar-Línea.
                [Selecciónar:  Usar método de búsqueda de
                          cadenas
                               Usar método de avanzar por
                          líneas]
            GOAL: Modificar texto
                     [Seleccionar: Usar método de Borrar palabra1
                                   Usar método de Borrar
                                   palabra2]
   Verificar Edición realizada
```

Figura 3.17: Corrección de un texto por medio de un editor de texto descrito mediante GOMS.

En el ejemplo, la tarea de introducir una serie de cambios en un texto en el ordenador a partir de unas correcciones en un manuscrito se realiza del siguiente modo. En primer lugar, tenemos un objetivo de nivel general superior que hace referencia al hecho de editar el manuscrito. Para realizar esta tarea se entra en un ciclo iterativo en el que primero se mira en el manuscrito cuál es la siguiente corrección (adquirir tarea unidad), y luego se realiza la corrección. Esta corrección necesita en primer lugar detectar la línea correspondiente (para lo cual hay dos métodos posibles) y luego realizar la modificación (para lo cual de nuevo hay dos métodos). Por último, se verifica si la edición ha sido realizada. Estos últimos pasos se ejecutan hasta que todas las modificaciones han sido realizadas.

A continuación daremos una introducción al otro modelo definido por Card, Moran y Newell denominado KLM (Keystroke Level Model).

---

[6] Desgraciadamente.

Para Green et. al (Green, et al., 1988), KLM responde a la estrategia más simple posible para modelar la dificultad de usar un interfaz: contar el número de acciones que el usuario necesita para ejecutar algunas tareas prototípicas. Si aceptamos que todas las tareas duran el mismo tiempo y que podemos ignorar el tiempo de pensar o planear, esto nos dará una predicción del tiempo relativo necesitado.

Una consecuencia de esta estrategia nos lleva a que en KLM no existe una estructura de control como en GOMS que progresivamente desgrane subobjetivos y métodos hasta alcanzar la tarea realizada. KLM es capaz de predecir tiempo de ejecución de un método, si se le proporciona como input los pasos que se van a seguir.

Este modelo se compone de varios operadores primitivos. Estos son:

K: Una pulsación en el teclado o apretar el botón del ratón. El tiempo asignado a una pulsación dependerá de la población de usuarios considerada. Los sujetos pueden variar hasta en un factor de 15 en esta característica.

P: Significa señalar a un objetivo por medio del ratón o otro mecanismo de ese tipo.

R: Tiempo que tarda el sistema en responder.

M: Tiempo del usuario empleado en prepararse para la tarea a continuación.

Apoyándose en estudios de Psicología Cognitiva de aquellos tiempos una serie de tiempos promedios son asignados a estos operadores para de ese modo poder realizar las predicciones correspondientes.

3.4.2.2.2. Criticas a GOMS y a KLM

Existen multitud de críticas que es posible realizar a estos modelos. No obstante, para ser justos, como comenta Gugerty (Gugerty, 1993), hay que hacer constar que la mayoría de estas críticas las lanzaron los propios autores desde el primer momento:

1) Estos métodos no son capaces de predecir los errores. KLM en absoluto, y GOMS, aunque tiene cierta capacidad, prácticamente igual. En realidad, esta afirmación, antes que una crítica es más apropiado decir que hace referencia a uno

de los supuestos establecidos por Card et. al proponer su modelo. Debido a que estos modelos se basan en una descomposición bien establecida de objetivos y subobjetivos, cuando los sujetos cometen errores, no es posible dar ningún tipo de explicación. Así. usualmente, los tiempos dedicados en cometer esos errores y más tarde en recuperarse de ellos en los experimentos analizados con estas notaciones es simplemente dejado de lado (Card, et al., 1983; de Haan, et al., 1991; Gugerty, 1993).

2) El modelo KLM no es capaz de representar la situación en la que diferentes métodos podrían ser aplicables en función de la situación concreta que se maneje. Es decir, no tiene reglas de selección que controlen cursos apropiados a seguir. Tampoco es capaz de considerar los aspectos físicos de las herramientas utilizadas, ni es fácil que pueda generalizar el modelo para comparaciones con sistemas parecidos (Green, et al., 1988).

3) En los trabajos originales acerca de GOMS hay un exceso de parámetros que se ajustan a las situaciones concretas de los experimentos que realizan. Por ejemplo, los métodos para realizar tareas no son postulados por las descripciones, sino que son extraídos de las ejecuciones de los usuarios estudiados. Los tiempos medios también son los provenientes de las propias tareas realizadas (Green, et al., 1988).

4) El concepto de tarea unidad o básica no está definido de manera satisfactoria. Para Card et. al, una tarea básica puede modificarse en función del contexto en que es realizada llevando a, por ejemplo, unir dos tareas en una sola. No hay una manera sencilla de entender o explicar esta situación (Green, et al., 1988). Una forma similar de realizar esta crítica es la siguiente (Gugerty, 1993), el tiempo que los operadores básicos necesitan para ser ejecutados es independiente del contexto y operadores iguales deberían de necesitar la misma cantidad de tiempo en momentos diferentes de la tarea y a través de tareas.

5) Muy a menudo, los usuarios no tienen un conjunto de procedimientos almacenados en memoria de una manera clara y jerárquica, tal y como postula GOMS (Gugerty, 1993; Whiteside y Wixon, 1987). Por el contrario, los usuarios pueden tener objetivos mal definidos que van tomando forma a lo largo de los diferentes estadios de la tarea. Esto se denomina planeamiento oportunista.

6) Las descripciones realizadas pueden convertirse en una técnica post-hoc que corre el riesgo de ser definida a partir del diálogo del ordenador antes que del

usuario. Sería necesario intentar desarrollar jerarquías naturales que fueran definidas según procedimientos preexistentes al sistema (Dix, et al., 1993).

7) El modelo en sus orígenes no era capaz de predecir aprendizaje ni tampoco recuerdo después de un periodo sin uso. Ni tampoco como diseñar un sistema que fuera fácil de aprender (Olson y Olson, 1990).

8) Los procesos cognitivos son tratados de una manera más superficial que los perceptivos y motores a la hora de predecir la conducta (Olson y Olson, 1990).

9) El modelo trata sobre todo tareas que son básicamente seriales, no siendo capaz de afrontar la situación en la que estas ocurran en paralelo (Olson y Olson, 1990).

10) El modelo no tiene en cuenta la carga mental (Olson y Olson, 1990).

11) No ofrece una manera de determinar la funcionalidad, es decir, las tareas que el sistema debe realizar. Esta debe ser determinada previamente a realizar un análisis GOMS (Olson y Olson, 1990).

12) No hay predicción de la fatiga de los usuarios, ni tampoco de las diferencias individuales entre éstos (Olson y Olson, 1990).

13) No hay manera de evaluar la satisfacción global del usuario (Olson y Olson, 1990).

14) El modelo no considera en absoluto como se comportará el sistema al ser incluido dentro de una organización (Olson y Olson, 1990).

15) Un último problema es que la notación utilizada para realizar las descripciones no estuvo muy bien establecida en ningún lugar. Solo las aportaciones posteriores (NGOMS fundamentalmente) proporcionaron guías concretas para la realización de estas descripciones por lo que no hay información para ayudar a solucionar los problemas prácticos que a menudo aparecen al realizar las descripciones.

Como señala Gugerty (Gugerty, 1993), muchas de estas críticas son injustas ya que enfrentan al modelo con críticas acerca de aspectos que nunca pretendieron cubrir y parece que se deben más al hecho de esperar demasiado de él. Además, muchas de las críticas pueden no ser validas en la actualidad, al haber sido

superadas por desarrollos posteriores. Veremos en el apartado siguiente, algunas de las aportaciones que han superado deficiencias en GOMS.

3.4.2.2.3. Elementos añadidos a GOMS.

1) La independencia de los elementos de GOMS del contexto en el que se producen: hasta la fecha, la mayoría de los estudios de predicción de GOMS parten de una serie de tiempos calculados a partir de los propios resultados obtenidos. Por ejemplo, en un estudio típico de Kieras y Polson, el número de producciones es utilizado para predecir tiempo de aprendizaje por medio de un análisis de regresión, siendo el efecto medio de cada regla obtenido por medio de esta técnica.

El trabajo de Olson y Olson (Olson y Olson, 1990), supuso un avance para superar este problema. A partir de una revisión de estudios anteriores obtuvieron tiempos de operador que utilizaron para predecir con relativa precisión tiempos de ejecución del usuario en otro estudio. No obstante, este éxito debería todavía repetirse en mayor número de estudios, estando todavía abierta la cuestión de la independencia de los tiempos de ejecución respecto del contexto (Gugerty, 1993).

2) El hecho de no poder afrontar tareas en las que los usuarios no tengan un conjunto de objetivos organizados jerárquicamente no puede ser considerada una crítica, ya que este modelo nunca pretendió cubrir ese aspecto de la interacción. Otros modelos (PUM por ejemplo, véase arquitecturas cognitivas), inspirados de una manera lejana en GOMS han ofrecido aportaciones a este problema.

3) La necesidad de seguir pasos estrictamente seriales en las descripciones GOMS ha sido tratada por medio de un modelo denominado CPM-GOMS (Gray, et al., 1993; Gray, et al., 1990). Este modelo combina GOMS con el análisis de pasos críticos (una técnica prestada de la ingeniería) y tiene de especial la capacidad de tener en cuenta componentes cognitivos, perceptivos y motores que se solapan parcial o totalmente en el tiempo. Esta modificación de GOMS produjo uno de los resultados más espectaculares en el campo de la interacción hombre-ordenador al ser capaz de predecir un ahorro de 3 millones de dólares anuales a una empresa telefónica si la vieja tecnología era mantenida en lugar de incorporar la nueva tecnología. Datos empíricos corroboraron estos resultados en un estudio de cuatro meses de duración.

4) La capacidad de predecir el tiempo de aprendizaje ha sido tratado por Kieras y Polson por medio de un modelo denominado CCT (Cognitive Complexity

Theory) y será tratado de modo extenso en un apartado posterior. Este apartado incluye también trasferencia de aprendizaje y otras cuestiones.

5) La carga mental ha sido medida en un estudio realizado por Lerch et. al. (Lerch, et al., 1989) en el que se comprobó que los errores realizados escribiendo fórmulas en una hoja de cálculo podrían ser predichos a partir de una medida de la carga mental derivada de un análisis GOMS. Otro estudio (Smelcer, 1989; citado en Olson y Olson, 1990), obtuvo resultados similares realizando preguntas a una base de datos utilizando el lenguaje SQL. No obstante, en ninguno de estos estudios se clarifica si fue el "pico" de elementos en memoria de trabajo la variable independiente, el nivel medio, o la duración temporal de esos ítems en la memoria.

3.4.2.3. CCT (Cognitive Complexity Theory) de Kieras y Polson

El objetivo del trabajo de Kieras y Polson es llevar a cabo una descripción de la complejidad de los sistemas o dispositivos desde el punto de vista de los usuarios. Denominan a su modelo Teoría de la Complejidad Cognitiva (CCT). Para ello se plantean llevar a cabo en primer lugar una descripción del conocimiento que los usuarios necesitan para usar el dispositivo y en segundo una descripción del comportamiento del dispositivo. Estas dos representaciones, combinadas con técnicas de simulación son de gran valor al permitir predecir una serie de parámetros psicológicos tales como la productividad o el tiempo de aprendizaje.

3.4.2.3.1. El modelo teórico-general.

El modelo afronta, en primer lugar, la representación del conocimiento necesario para usar el dispositivo, y, en segundo, la descripción del comportamiento del dispositivo. A partir sobre todo de la representación del conocimiento, el modelo realiza una serie de predicciones acerca del tiempo de aprendizaje y el de ejecución.

El conocimiento para usar el dispositivo.

El supuesto básico del modelo es que el contenido, estructura y cantidad de conocimiento requerido para ejecutar un sistema determinan tanto la productividad como el tiempo de aprendizaje (Polson, 1987). Su modelo descansa en los siguientes supuestos:

1) Contenido del conocimiento: Kieras y Polson (Polson, 1987) asumen que el conocimiento necesario para manejar un dispositivo se compone de tres elementos: el conocimiento de las tareas que pueden ser ejecutadas por el sistema, el conocimiento de como llevar a cabo los procedimientos necesarios para completar cada tarea y la representación de como funciona internamente el dispositivo (es decir, el modelo mental -Norman, 1986-).

El conocimiento de como llevar a cabo las tareas es descompuesto en las cuatro categorías del modelo GOMS (Card, et al., 1983).

2) La estructura del conocimiento: siguiendo la propuesta de GOMS, CCT descompone las tareas complejas en una secuencia de subtareas. Así, cada tarea o subtarea tiene un objetivo o subobjetivo asociado que indica la intención de llevarla a cabo. La descomposición es representada como una estructura de objetivos que define las interrelaciones entre las subtareas y especifica la secuencia de subtareas necesarias para llevar a cabo la tarea.

3) Conocimiento específico de la tarea y del interface: dependiendo del nivel al que se encuentren los objetivos, más probablemente dependerán de un tipo de conocimiento u otro. Así, los objetivos de nivel más superior están relacionados con la tarea a llevar a cabo, mientras que los subobjetivos de un nivel más inferior estarán más relacionados con los detalles particulares del sistema que se esté tratando.

4) La cantidad de conocimiento: la cantidad de conocimiento necesario para poder manejar un dispositivo es un valor obtenido ejecutando una simulación de un modelo de como llevar a cabo las funciones. Esta simulación permite calcular valores para el tiempo de entrenamiento y la productividad. Esta simulación está basada en el concepto de sistema de producción y será revisada posteriormente.

5) La representación en términos de sistemas de producción: Kieras y Polson (Bovair, et al., 1990; Kieras y Polson, 1985; Polson, 1987) representan el modelo CCT en términos de un sistema de producción. Así,

a) los objetivos son representados directamente, y aparecen en las condiciones de las reglas,

b) los métodos son representados como secuencias de reglas cuyo primer miembro es disparado por la aparición del objetivo para hacer el método.

c) las reglas de selección son representadas como reglas disparadas por un método general y un contexto específico que indica el objetivo para llevar a cabo el método concreto

d) los operadores consisten de acciones elementales por un lado y otras más complejas que realizan evaluaciones del entorno por otro (como por ejemplo mirar si un elemento está presente, etc.).

En este modelo no hay representación de procesos cognitivos complejos, tales como la comprensión lectora o procesos perceptivos. En su lugar, se sustituyen estos procesos por un operador especial-complejo tal como `Leer-del-manuscrito`. Este hecho se justifica en que tratar con la representación de esos procesos aumentaría enormemente la complejidad de los modelos sin añadir aspectos relevantes al diseño del interface.

El sistema de producción concreto utilizado por los autores se denomina sistema de producción parsimonioso (PPS) y presenta las siguientes características:

Una regla PPS de producción contiene cinco términos encerrados entre paréntesis:

(<nombre> IF <condición> THEN <acción>)

Estos términos se refieren a:

Nombre: Es una etiqueta sin valor funcional. Sólo existe para facilitar las tareas de depuración y programación.

Condición: La condición de una regla es una lista de sentencias que deben ser todas verdaderas para que la condición de la regla sea verdadera. Dentro de esta lista de condición se admite el modificador NOT para comprobar la ausencia de una determinada condición y el prefijo ?, el cual índica que el valor a emparejar es una variable en lugar de una constante.

Acción: Las acciones son secuencias de operadores que pueden ser internas (añadiendo o eliminando objetivos y notas en la memoria de trabajo) las cuales generalmente preparan la ejecución de otra regla en el siguiente ciclo, o externas tal como apretar una tecla o mover el ratón. También, las acciones pueden ser operadores complejos tales como Leer-del-manuscrito.

El funcionamiento es el usual en un sistema de producción. El interpretador explora la memoria de trabajo en busca de reglas que puedan ser disparadas. Ello hace que la parte de acción sea ejecutada realizando modificaciones en la propia memoria de trabajo o en el contexto de la tarea. En caso de existir conflictos entre reglas, el mecanismo no tiene forma de solucionarlo, y todas las reglas cuyas condiciones hayan sido satisfechas. Esto significa que las reglas deberán ser escritas de tal modo que no sean disparadas repetidas veces, ya que el interpretador no posee ninguna manera de controlar este suceso. Esto lleva a que el conocimiento de control es totalmente explícito, y no hay ningún conocimiento implícito en el interpretador.

La representación del comportamiento del dispositivo.

Esta notación entra dentro de la clasificación de representaciones del sistema y está dirigida a crear un modelo del comportamiento del dispositivo que responda a los inputs del usuario. Esta notación está basada en lo que ellos denominan una red de transición generalizada (Generalized Transition Network), la cual tuvo una importancia relativa en las primeras presentaciones de la teoría (Kieras y Polson, 1985) pero que no se ha mantenido en las presentaciones posteriores.

El modelo de trasferencia-aprendizaje.

Un elemento que incorpora CCT y que no había sido contemplado por GOMS es un modelo simplificado de aprendizaje (Bovair, et al., 1990; Kieras y Polson, 1985; Kieras y Bovair, 1986). En él, el tiempo necesario para aprender una nueva función de un dispositivo está relacionado linealmente con el número de reglas nuevas que es necesario aprender para manejar esa función y no el número total de reglas.

Para calcular el número de reglas nuevas, Kieras y Polson asumen un modelo de trasferencia simple semejante a la teoría clásica de elementos comunes de Thorndike en la que los elementos comunes son reglas de producción. Así, una regla puede, o bien ser completamente nueva, o ser idéntica, o ser generalizable a partir de otra ya conocida previamente. En los dos casos últimos se asumiría que el sujeto no invierte ningún esfuerzo en su aprendizaje.

El modelo de ejecución.

El modelo de como los sujetos ejecutan las tareas consiste únicamente en llevar a cabo una simulación del modelo previamente descrito. Esta simulación proporciona las siguientes variables predictoras: número de ciclos de reconocimiento-acción y el número y tipo de acciones necesitadas para ejecutar cada tarea. La ejecución sobre una tarea es calculada en términos de tiempo utilizando la siguiente fórmula:

```
Tiempo_de_ejecución=nº_de_ciclos X tiempo/ciclo + tiempo_invertido_en operadores
```

Un inconveniente de este modelo es que en los artículos publicados por los autores, el tiempo correspondiente a cada operador debe ser estimado a partir de los datos mediante técnicas de regresión. Ello no obstante choca con la sugerencia de Card et. al (1983) de que este tipo de valores podrían ser calculados independientemente de modo que los desarrolladores pudieran utilizarlos para calcular el tiempo de ejecución sin haber llevado a cabo ningún experimento. Siguiendo esta idea, Olson y Olson (Olson y Olson, 1990), recogiendo diversos estudios, avanzan una serie de valores para operadores habitualmente utilizados en análisis GOMS y muestran que existe una constancia en ellos. Además utilizando valores de este tipo realizaron predicciones de experimentos independientes y encontraron un promedio de error del 14%.

3.4.2.3.2. La notación.

A continuación se presenta un resumen de las convenciones y reglas de estilo que los autores de CCT proponen a la hora de realizar sus descripciones (Bovair, et al., 1990). Estas reglas de estilo son constricciones que deberían impedir a las reglas de producción comportarse de una manera completamente arbitraria.

Convenciones notacionales.

Convenciones para sentencias de condiciones: Las sentencias de condiciones tienen la forma `(Etiqueta Término1, Termino2,…,Terminok)`. Por convención, el número de términos que sigue una etiqueta es fijo. Por ejemplo, dos términos siguen a la etiqueta `OBJETIVO`, y tres siguen a la etiqueta `DISPOSITIVO`.

Convenciones para etiquetas de sentencias: Hay cuatro tipos de etiquetas utilizadas normalmente en los análisis CLG. `OBJETIVO` hace referencia a una sentencia relacionada con los objetivos usados en GOMS. `NOTA` es usado para

información que debe ser mantenida en la memoria de trabajo sobre el transcurso de varias reglas, lo cual puede proporcionar un contexto específico para un método o para controlar la ejecución. PASO es usado para los "pasos objetivo" los cuales controlan la secuencia de ejecución dentro de un método. DISPOSITIVO indica la información que es proporcionada directamente por el dispositivo.

Convenciones para sentencias de objetivos: El formato de una sentencia de OBJETIVO es (OBJETIVO Término1 Término2). Término1 es el verbo o acción del objetivo y Término2 representa el objeto del verbo o acción. Así, el objetivo de borrar un carácter es representado como (OBJETIVO BORRAR CARACTER).

Reglas de estilo

Generales

Regla 1: Las reglas de producción deberían generar la secuencia correcta de acciones del usuario.

Regla 2: La representación se ajustaría a principios de programación estructurados. De este modo, la representación debería escribirse de arriba-abajo y las rutinas llamadas de una manera standard, no afectar el contexto del que ha hecho la llamada y borrar la información local que usan o crean.

Regla 3: La representación de nivel superior para muchas tareas debería tener una estructura de tarea-unidad.

Regla 4: La estructura de tarea-unidad de nivel superior debería estar basada sobre el objetivo de nivel superior. El objetivo de nivel superior es uno del tipo "editar manuscrito" o "realizar análisis". Este objetivo determina una tarea-unidad con esta estructura: buscar la siguiente tarea que se dirija a la consecución de ese objetivo, añadir el objetivo de ejecutarla y, si no hay más tareas que realizar, detener el sistema.

Regla 5: Las reglas de selección determinan el método a aplicar. Dado un OBJETIVO específico, y un contexto dado, las reglas de selección determinan que método llevar a cabo.

Regla 6: La información necesitada por un método debería ser proporcionada a través de la memoria de trabajo.

Regla 7: La secuenciación dentro de un método es mantenida encadenando Pasos. Al ejecutar un método hay una serie de pasos que deben ser ejecutados por orden. Cada método tiene una regla de comienzo que indica el primer PASO a llevar a cabo. Cuando la regla correspondiente es disparada ello lleva a un borrado del PASO anterior y al añadido de otro PASO que disparará el siguiente elemento en el método. Al terminar la secuencia, la regla final borra el objetivo que se quería llevar a cabo y añade una NOTA indicando que el método ha terminado.

Regla 8: Las etiquetas para los PASOS deberían estar basadas en las acciones que llevan a cabo las reglas y ser usadas consistentemente.

Regla 9: Usar una NOTA que bloquee una regla durante el tiempo que el método correspondiente esté en progreso. Ello evita que esta regla sea continuamente disparada por el objetivo actualmente en memoria de trabajo. Para ello, la condición de la regla probará por la ausencia de esa nota antes de dispararse.

Representando distintos grados de conocimiento.

En esta sección se muestran reglas de estilo para describir dos distintos grados de conocimiento. Estos grados se denominan el novato, que corresponde a aquel usuario que conoce todos los métodos del sistema y es capaz ejecutarlos correctamente pero no ha tenido oportunidad de practicarlos de un modo extenso, y el usuario con práctica, el cual corresponde aproximadamente al usuario experto.

Representando al usuario novato.

Las reglas de estilo para los usuarios novatos son:

Regla Novata 1: Cada acción visible requiere una regla de producción separada. Manipulaciones de la memoria de trabajo tales como borrar y añadir notas no son consideradas pasos separados y por tanto puede haber varios en una sola regla.

Regla Novata 2: Los novatos comprueban explícitamente todo el feed-back del sistema.

Representando al usuario con practica.

Con la práctica lo que cambian son las reglas, no la velocidad de ejecución de las reglas. Esto significa que la práctica cambiará la representación de las reglas y estos cambios se reflejarán en el número y contenido de las reglas.

Hay dos posibles direcciones teóricas que podrían guiar estas transformaciones. Una de ellas implicaría la aparición de nuevos métodos más especializados para llevar a cabo las tareas. El otro, el colapsamiento de reglas mediante, por ejemplo, el proceso de composición (Neves y Anderson, 1981). En CCT se asume que la práctica compacta el conjunto de reglas antes que genera nuevos métodos especializados. Los autores presentan un algoritmo para llevar a cabo esta compactación, pero de modo resumido, el efecto de este algoritmo es el siguiente:

1) Eliminar las comprobaciones explícitas de todo el feed-back del sistema.

2) Las reglas que se ejecuten siempre en un orden fijo deberían ser compuestas en una regla simple.

3) Las operaciones complejas de los métodos serían perfeccionadas por medio de la práctica, de modo que los usuarios expertos podrían llevarlas a cabo más rápidamente, disminuyendo por tanto el tiempo estimado para el operador.

Notaciones simplificadas: NGOMS

Kieras (Kieras, 1988) propuso una notación que facilitara el trabajo de construir reglas de producción, ya que escribir éstas podría considerarse análogo en muchos casos a programar en lenguaje ensamblador. En nuestro trabajo, hemos optado por utilizar esta notación.

En este trabajo se presentan fundamentalmente dos cosas. En primer lugar, una serie de convenciones en lenguaje natural acerca de la nomenclatura a utilizar en un análisis NGOMS, así como una serie de operadores (tanto externos como internos) considerados más importantes. En segundo lugar, se describe un procedimiento general que permitiría la construcción de los modelos NGOMS. Este procedimiento puede verse como un esquema general en el que se irían sustituyendo los detalles particulares de los diseños concretos llevados a cabo (véase capítulo 3 para un ejemplo y algunos componentes suplementarios a la descripción que sigue).

La notación es como sigue (Kieras, 1991):

1) Objetivos: Un objetivo es un par acción-objeto tal y como Borrar-Objeto. El autor especula acerca de la diferencia entre objetivos específicos del dispositivo (marcados por la estructura del dispositivo) e independientes del dispositivo.

2) Operadores: Los operadores son acciones que ejecuta el usuario. La distinción entre un operador y un objetivo no es sin embargo tan obvia como puede parecer y es más bien relativa: depende del nivel del análisis. La idea de los análisis NGOMS se basa en que primero se indican una serie de operadores de muy alto nivel y, posteriormente, esos operadores se sustituyen por objetivos que deben ser realizados por operadores a un nivel inferior. Existen varias clases de operadores:

a) Operadores externos: Son las acciones observables a través de las cuales el operador intercambia información con el sistema u otros objetos en el contexto. Pueden ser operadores que obtienen información o acciones que producen un resultado.

b) Operadores mentales: Son acciones internas ejecutadas por el usuario, que son no observadas e hipotéticas, inferidas por el teórico o el analista. Son por ejemplo, tomar una decisión, recuperar un ítem de la memoria de trabajo, etc. Hay operadores mentales elegidos para representar tareas de alto nivel: son operadores definidos por el analista y cumplen la función de sustituir una operación compleja que por una razón de desconocimiento o de conveniencia se prefiere no intentar describir.

c) Operadores primitivos y de alto nivel: Un análisis de tareas se realizará a un determinado nivel, el cual es considerado el "grano" del análisis. Si un operador no es descompuesto hasta el nivel inferior aunque podría serlo, entonces se trata de un operador de nivel superior. Si no puede ser descompuesto, entonces es un operador primitivo.

Los operadores primitivos externos establecidos son:

```
Mover-mano-al-ratón
Presionar-tecla (nombre de tecla)
Teclear <cadena de caracteres>
Mover-cursor-a <destino>
Encontrar-cursor-en <devolver coordenadas cursor>
Encontrar-menu-item <descripción-item-menú>
```

Los operadores primitivos mentales son:

- Flujo de control

```
Llevar a cabo el objetivo de <descripción de objetivo>
```

Esto equivale a llamar a una subrutina.

```
Devolver con objetivo realizado.
```

Es equivalente a una sentencia de Return

```
Decidir: Si <operador…> entonces <operador…> sino <operador…>
```

Este operador es para decisiones simples. Hay que recordar que el sistema admite reglas de selección entre métodos cuando es necesario.

```
Ir a paso<número>
```

Este operador debería ser utilizado lo menos posible. Esta es una recomendación usual también cuando se habla de lenguajes de programación.

- Almacenamiento en memoria y recuperación.

```
Recuperar que <descripción de objeto en WM>
Retener que <descripción de objeto en WM>
Olvidar que <descripción de objeto en WM>
Recuperar-MLP que <descripción objeto en MLP>
```

Estas operaciones significan: Traer información que está actualmente en la memoria de trabajo para ser utilizada, almacenar información para posteriormente utilizarla, desechar un ítem como importante para ser recordado (y por tanto susceptible de ser olvidado) y recuperar un ítem de la memoria a largo plazo.

- Los operadores mentales definidos por el analista son:

```
Obtener de la tarea <nombre>: Mira información en la descripción de la
tarea etiquetada con <nombre> y la almacena en la memoria de trabajo.
Verificar resultado: Determina si el resultado obtenido es el deseado.
Obtener-siguiente-localización-tarea: Obtiene donde realizar la
siguiente tarea.
```

4) Métodos: Un método es una secuencia de pasos que lleva a cabo un objetivo. Un paso es bien un operador externo primitivo, bien un operador mental que sitúa y lleva a cabo un objetivo.

La forma para un método es la siguiente:

```
Método para llevar a cabo el objetivo de <descripción del objetivo>
Paso 1. <operador>
Paso 2. <operador>
…
Paso n. Devolver con objetivo realizado.
```

A menudo los métodos incorporan como pasos llamadas a otros métodos, componiendo la jerarquía de métodos y submétodos que es la verdadera descripción NGOMS.

5) Reglas de selección. El propósito de una regla de selección es dirigir el control hacia el método apropiado para llevar a cabo el objetivo. Esto sólo es posible, naturalmente, cuando más de un método está disponible para llevar a cabo el objetivo. La forma general de especificación de las reglas de selección es la siguiente:

```
Regla de selección para el objetivo de <descripción general del objetivo>
Si <condición> entonces llevar a cabo objetivo de <descripción específica del objetivo>
Si <condición> entonces llevar a cabo objetivo de <descripción específica del objetivo>
…
Devolver con objetivo realizado
```

A partir de estos elementos, Kieras (1991) proporciona una serie de claves para realizar el recuento de reglas de producción. En breve, estas son:

1) Cada paso cuenta como una regla.

2) La sentencia de método también cuenta como una regla.

3) Cuando el paso incluye un Decide, éste cuenta como un paso sino incluye un sino, y como dos reglas cuando sí incluye un sino.

4) La sentencia de regla de selección cuenta como una regla.

5) La sentencia de devolver con objetivo realizado cuenta como una regla.

3.4.2.3.3. Trabajos empíricos.

Los trabajos empíricos realizados con NGOMS o CCT se han centrado en las predicciones de tiempo de aprendizaje. Esencialmente, sus estudios consistieron en hacer que usuarios aprendieran una serie de tareas, variando el orden en que estas fueron aprendidas. Se registraba el tiempo empleado en aprender estas tareas y se intentaba predecir el tiempo a partir de modelos basados en GOMS. El número de reglas nuevas en cada procedimiento era calculado y se introducía en un análisis de regresión con objeto de examinar si eran capaces de explicar un porcentaje de varianza significativo. Otras variables de control también utilizadas en los análisis eran:

• Número de reglas totales: Que mide fundamentalmente la longitud de los métodos.

• Número de serie en la que el procedimiento era aprendido.

• Tiempo medio que los sujetos emplearon en aprender los métodos. Esto es fundamentalmente una manera de manejar el diseño intrasujeto (Pedhazur, 1982)[7]. Esto se justifica al no tener NGOMS ningún mecanismo para predecir las diferencias individuales en aprendizaje[8].

• Grupo según la seriación con la que los sujetos aprendieron los procedimientos.

• Variables ficticias para el primer procedimiento, debido a que en general significó un tiempo especialmente grande en comparación con otros procedimientos.

---

[7] En el vocabulario econométrico esto sería un modelo de regresión con variables ficticias para los efectos individuales (Maddala, 1985), ya que codificar por medio de una variable con los efectos medios o por medio de variables ficticias (Pedhazur, 1982) es equivalente.

[8] Una justificación similar a la simbolizada por la expresión "ignorancia específica" indicada por las variables ficticas frente a la "ignorancia generalizada" del término de error (Maddala, 1971; Maddala, 1987a).

Siguen a continuación algunos ejemplos de trabajos experimentales diseñados para comprobar las predicciones realizadas tanto en cuanto a tiempo de aprendizaje como en ejecución.

Aprendizaje

En Polson y Kieras (1984) se encontró que el número de nuevas producciones era capaz de explicar el 61% de la varianza.

Hasta ahora, la mayor parte de los trabajos de validación de CCT provienen de sus propios autores. Así, Kieras y Bovair (Kieras y Bovair, 1986) encontraron que el número de reglas nuevas en cada procedimiento de un dispositivo era capaz de explicar por sí solo el 69% de la varianza en el tiempo de aprendizaje y el 41% cuando otras variables tales como el número de reglas totales, las media individuos para todas las tareas, las reglas generalizadas, etc. fueron introducidas en el análisis de regresión.

En Polson (1987) los resultados fueron todavía más precisos, explicando un modelo, basado únicamente en el número de reglas nuevas, entre un 81%, 88% y un 90% de tres tareas de edición de textos. En este caso se utilizaron las medias de los sujetos para cada procedimiento.

No obstante, en Bovair, Kieras y Polson (1990), a pesar de que el número de reglas nuevas siguió siendo el mejor predictor del tiempo de aprendizaje sus efectos descendieron con respecto al estudio anterior pasando a explicar un 33% de la varianza. En este caso se utilizó la media total de los sujetos como variable predictora.

Ejecución

En Polson (1987) se comprobaron las predicciones de ejecución para sujetos recién entrenados (novatos) y expertos. Dos descripciones distintas (una para novatos y otra para expertos) fueron hechas y se ajustaron a los datos de unos sujetos en una tarea de edición de textos. De nuevo, promediando los resultados de los individuos a través de tareas, la ecuación encontrada explicó un 80% de varianza de los datos, utilizando como variables predictoras el número de ciclos acción-reconocimiento de la simulación y el número total de notas en la memoria de trabajo. Los modelos de experto y novato elaborados según mecanismos de

composición hicieron mejores predicciones que otros modelos que asumieron un decrecimiento del tiempo de ejecución por cada paso.

En Bovair, Kieras y Polson (1991) se planteó una predicción de los datos de ejecución, promediados a través de sujetos, a partir de una serie de variables extraidas de la simulación, hallando los siguientes resultados:1) Las coeficientes atribuidos a las verificaciones hechas por novatos y expertos seguían las predicciones hechas por la teoría., 2) el número de ciclos en la simulación siguió siendo un predictor de gran importancia y 3) el número de notas y carga en la memoria de trabajo no fue un buen predictor (explicado porque la tarea utilizada no sobrecargaba en exceso ésta).

3.4.2.3.4. El uso de CCT en situaciones de diseño.

Karat y Bennet (Karat y Bennett, 1991) presentan un ejemplo en el que muestran como las técnicas de CCT pueden ser utilizadas en situaciones de diseño real, afrontando por tanto el verdadero objetivo para las que fueron diseñadas. En este capítulo muestran como de dos procedimientos relativamente similares (llevar a cabo una copia de disco en los entornos Macintosh y Windows), uno de ellos presenta problemas de inconsistencia (Macintosh) y el otro necesita una excesiva carga de memoria (recordar el path hacia el destino).

Algunas opiniones de los autores son las siguientes: 1) la falta de un método para llevar a cabo el análisis de tareas es una deficiencia grave, 2) a menudo, aunque no hubiera pruebas científicas de los beneficios de esta orientación, los diseñadores vieron con claridad problemas en los ejemplos descritos y 3) quizás buena parte de las intuiciones obtenidas provengan simplemente de la experiencia de los diseñadores y no de una propiedad de CCT. No obstante, el tener una metodología común facilitó la discusión y confrontamiento de los posibles problemas de diseño.

Elkerton y Palmiter (Elkerton y Palmiter, 1991) utilizaron GOMS para diseñar un sistema de ayuda para un programa, así como documentación escrita.

Se trata de un estudio que comparó la creación de un sistema de ayuda para Hypercard a partir de una descripción basada en NGOMS. Para ello se desarrolló una pila (un documento de base de datos con propiedades de hipertexto) en Hypercard que sustituía a la originalmente proporcionada por los desarrolladores

comerciales. Los resultados en líneas generales aportaron evidencia de que el nuevo sistema de ayuda tuvo un comportamiento superior al original.

Irving et. al (Irving, et al., 1994) realizaron un estudio del interfaz utilizado para manejar un buen número de tareas en una cabina de piloto. Los resultados mostraron como buena parte de las tareas necesitaban una serie de procedimientos "centrales" que enlentecían considerablemente el tiempo de aprendizaje y ejecución y que por lo tanto ofrecían una oportunidad para un rediseño.

Gong y Kieras (Gong y Kieras, 1994) estudiaron la reconversión de un programa originalmente diseñado para el entorno DOS al entorno Macintosh. El análisis GOMS permitió una serie de modificaciones al interfaz que hicieron mejorar en un 46% como promedio el tiempo de ejecución empleado por sujetos experimentales con respecto a una primera versión del programa no analizada mediante GOMS. Las predicciones realizadas se aproximaron con gran precisión en cuanto a las diferencias relativas entre interfaces. No obstante, las predicciones en tiempos absolutos no fueron tan precisas, siendo el doble en muchas casos a los tiempos originales. Una serie de modificaciones fueron necesarias en el procedimiento sugerido por Kieras (1991) para corregir estas diferencias como fueron: 1) Los tiempos sugeridos por Kieras son excesivos, pudiéndose reducir por un factor de 5. 2) Ciertos operadores son ejecutados en paralelo, en lugar de serialmente por lo que sus tiempo pueden ser reducidos. Esto ocurría en tres ocasiones: 1) En el operador perceptivo "localizar" un menú que se solapaba con un movimiento de ratón inmediatamente. 2) "Verificar selección" que se solapaba con la ejecución de la siguiente acción de teclado o ratón y 3) el uso del teclado o el ratón mientas la pantalla estaba siendo actualizada. Después de estas correcciones, las predicciones cayeron en un margen cercano en un promedio de un 9% a los resultados empíricos.

3.4.2.4. Extensiones a GOMS-CCT: Interfaces "de aprendizaje fácil ". El modelo CE+

El problema de analizar interfaces diseñados para ser utilizados sin ningún conocimiento previo es de gran interés, ya que un buen número de aplicaciones actuales se enfrentan a este desafío. Algunos ejemplos son: vendedores automáticos de billetes, cajeros automáticos y sistemas de información. Ahora bien, cada vez más, incluso interfaces a aplicaciones de mediana o gran

complejidad intentan alcanzar ese objetivo pretendiendo ofrecer un entorno en el que el usuario sepa inmediatamente qué hacer.

CE+ (Polson y Lewis, 1990) es uno de los modelos que intenta ofrecer un método y/o notación para explicar/analizar este tipo de interfaces. Otros modelos son el de Howes (Howes, 1994) y D-TAG (Howes y Payne, 1990) revisados anteriormente en relación con TAG.

El punto de partida de CE+ son tres fuentes que por separado no llegan a ser fácilmente aplicables, razón por la cual es necesario un nuevo modelo.

• 1) GOMS-CCT presenta el inconveniente de que asume que la dificultad de un interface está determinada fundamentalmente por la cantidad de conocimiento que es necesario para llevar a cabo una serie de métodos bien aprendidos. Este supuesto no parece aplicable al caso de los interfaces "de aprendizaje fácil " ya que en ellos la cantidad de conocimiento total que es necesario para utilizarlos es relativamente pequeño y además la situación en la que el usuario conoce bien los métodos no es la más crítica. Por el contrario, el proceso de solución de problemas y la forma en que éste se entrelaza con el aprendizaje y la propia ejecución es el punto crítico en este tipo de interfaces, y en él parece más importante la claridad de las indicaciones en la pantalla y el feed-back.

• 2).EXPL (Lewis, et al., 1987) es un modelo diseñado para explicar el papel del feed-back en el aprendizaje de procedimientos. Un análisis basado en este modelo intenta encontrar elementos que conecten las acciones ejecutadas por el usuario y las respuestas producidas por el sistema, estableciendo una conexión causal cuando lo logra. De este modo EXPL proporciona una guía complementaria a los análisis GOMS o CCT ya que muestra la importancia del feed-back, el cual es clave para determinar las conexiones. El inconveniente de este modelo es que no tiene ningún mecanismo que explique la forma en que el usuario analizar las acciones apropiadas a partir de las alternativas disponibles. Además, EXPL presupone ejemplos de los que el usuario aprende una conexión fallando en explicar la situación en la que el usuario advierte la acción correcta sin haberla ejecutado previamente.

• 3) Solución de problemas en dominios no familiares. Partiendo del concepto de espacio de problemas y de métodos de búsqueda (Newell y Simon, 1972a), Polson y Lewis examinan dos variantes del método de solución de problemas basado en medios-fines (el más común) usualmente hallados en estudios de

solución de problemas en novatos. Estos son "hill-climbing" (el sujeto sólo considera acciones legales y se selecciona la operación que ofrece el progreso mayor en dirección al objetivo utilizando la similaridad perceptual como medida de distancia respecto al objetivo) y "back-chaining" (el sujeto anticipa la solución que lograría el objetivo, aunque no sea posible ejecutarla, y una vez hallada intenta encontrar la acción que le permitirá llevar a cabo aquélla, siguiendo hasta hallar una acción legal en el espacio de tareas actual). De estos dos métodos, no obstante, ninguno es aplicable fácilmente al contexto del uso de interfaces "de aprendizaje fácil ". El "back-chaining" necesita conocimiento que probablemente no está disponible la primera vez que se utiliza un interface, mientras que "hill-climbing", aun siendo más aplicable, está limitado ya que los usuarios no pueden anticipar en muchas ocasiones los resultados de las acciones que ejecuten. Así, el método preferido es el denominado "label-following" el cual es presentado como un caso especial de la estrategia de "hill-climbing". Esta estrategia asume que las descripciones que comparten términos con los resultados deseados estarán probablemente conectados causalmente con ellos. Resultados experimentales muestran el uso por parte de los usuarios de esta estrategia, la cual puede ser relacionada con el hecho de que su conducta debe estar en su mayor parte determinado por la apariencia y contenido de la pantalla.

3.4.2.4.1. La descripción del modelo.

CE+ incorpora simultaneamente aprendizaje, ejecución y solución de problemas. Más en concreto, cada uno de estos procesos sigue la siguiente dinámica.

1) Solución de problemas.

El componente de solución de problemas de CE+ usa el método "label-following", es decir, cuando se encuentra ante situaciones alternativas escoge aquella cuya descripción se aproxima más al objetivo buscado a condición de que no lo haya intentado anteriormente. Más en concreto se siguen los siguientes pasos:

a) Se busca serialmente una acción cuya descripción se aproxime más que cierta cantidad umbral al objetivo deseado. Si una acción así es encontrada, se ejecuta con una probabilidad superior, si no ha sido ejecutada anteriormente, y con una probabilidad más baja si ha sido intentada anteriormente.

b) Si en el primer paso no se selecciona ninguna acción, en el segundo se ejecuta una acción al azar, si hay alguna que no haya sido intentada anteriormente.

c) Si no hay acciones no intentadas se aplican una de estas dos posibilidades. Si hay tres o menos alternativas se examinan todas las alternativas para determinar aquélla con mayor grado de solapamiento. Si hay más de tres alternativas una de las opciones es elegida al azar.

2) El componente de aprendizaje.

Una vez el componente de solución de problemas selecciona una acción para ejecutar, el modelo CE+ aprende la conexión entre la acción y el resultado y la almacena, sea o no acertada. De este modo el aprendizaje incidental es contemplado dentro de este modelo. Si la acción es acertada (juzgando tal cosa por la existencia de claves similares en el resultado a las presentes en el objetivo deseado) CE+ recuerda esto y lo mantiene disponible para el momento en que en el futuro desee alcanzar este objetivo de nuevo.

3) El componente de ejecución.

CE+ ejecuta siempre aquellas acciones que aprendió en el pasado y sea posible aplicar en la situación actual. Sólo cuando no es posible llevar a cabo ninguna acción se invoca el componente de solución de problemas.

3.4.3. Modelado cognitivo de los errores

Aunque otros modelos discutidos anteriormente podrían incluirse dentro de este apartado (en concreto Ayn de Howes y CE+ de Polson) ya que ambos comparten el interés por modelar el comportamiento de los usuarios no sólo libre de errores sino incluyendo éstos hemos preferido mantenerlos cerca de sus orígenes intelectuales dejando para esta última parte dos modelos que proponen ideas mucho más novedosas y radicales que aquellos. Los modelos que consideraremos son ICS de Barnard et. al. y PUM de Young et. al.

3.4.3.1. ICS (Interactive Cognitive Subsystems) de Barnard et. al.

Barnard (Barnard, 1988) postula como principios generales de su arquitectura los siguientes puntos: 1) la percepción, la cognición y la acción pueden ser analizados en términos de módulos de procesamiento de la información bien

definidos y discretos, 2) asume una separación entre las representaciones mentales de la memoria específica y los procesos específicos de contenido que manipulan esas representaciones, y, 3) existe por parte del sistema cognitivo humano una capacidad significativa para el procesamiento paralelo de la información.

En línea con otros modelos de la actividad mental, Barnard propone tres niveles de subsistemas dentro de la arquitectura cognitiva.

• Subsistemas sensoriales. Dentro de este nivel Barnard postula el subsistema acústico y el visual como los más importantes en HCI.

• Subsistemas representacionales: Son los encargados de la manipulación del input. Dentro de él encontramos el subsistema Morfonoléxico (encargado de la manipulación de la estructura externa del lenguaje), el subsistema Objeto (encargado del procesamiento de estructuras visoespaciales), y el Proposicional y el Implicacional (que mantienen el procesamiento de representaciones conceptuales y semánticas).

• Subsistemas efectores: Produce un output desde los sistemas representacionales y controla los outputs motores y emocionales. Son el subsistema articulatorio y el límbico.

Todos los subsistemas comparten una estructura de procesos comunes. En primer lugar, existe un proceso primario "copiar" que sirve para crear un registro episódico de toda la información que entra en el subsistema en el propio código que ese subsistema es capaz de manipular. En segundo lugar, existen un conjunto de procesos secundarios paralelos que sirven para recodificar la información y que pueden ser concebidos como un sistema de producción independiente funcionalmente. Si un conjunto de condiciones es satisfecho en el patrón de input entonces el proceso correspondiente lo transformará en un output dado.

Como el mismo autor señala, la relevancia de un modelo de este tipo para el científico aplicado puede parecer oscura. No obstante, esta descripción permite el llevar a cabo análisis de tareas cognitivas, que, por un lado muestren los subsistemas implicados en la realización de una tarea y el flujo de pasos que, dependiendo del conocimiento de los usuarios o sus características necesitan, y, por otro lado, incorporar investigación empírica dentro de cada uno de los subsistemas, de modo que diagnósticos acerca de la posible ejecución de los sujetos sean posibles.

A continuación veremos un ejemplo de análisis de tareas cognitivas utilizando este modelo.

3.4.3.1.1. Un ejemplo de descripción.

Lo siguiente es un ejemplo de descripción tomado de Barnard (1988), en el que se muestra la descripción del proceso de cambiar un párrafo mediante un procesador de texto.

En primer lugar el texto es procesado a través del subsistema visual y es pasado al sistema objeto, el cual lo representa atendiendo a sus características principales. A continuación, otro proceso transforma esa representación al código morfonoléxico, el cual representa la estructura superficial del lenguaje, marcando los constituyentes superiores (palabras y frases) del texto. La información en este código es pasada al subsistema proposicional el cual interpreta el significado de la frase. Este trasvase se produce por medio de un buffer, lo cual garantiza un procesamiento más extendido a lo largo del tiempo. Asimismo, el sistema morfonoléxico y el proposicional pueden entrar en un bucle recíproco que contribuya al procesamiento en ambos códigos.

Comprender	Borrar	Escritura creativa
* VIS -> OBJ :: :: OBJ -> MPL:: (:: COPY MPL -> MPL -> PROP ::) buf (::PROP -> IMPLIC:: ::IMPLIC -> PROP::) n	(::PROP -> IMPLIC:: :: IMPLIC -> PROP::) n ::PROP->OBJ:: ::OBJ->LIM:: ::LIM->MOT* Más "feedback" por *VIS->OBJ:: ::OBJ-<PROP::	(::PROP->IMPLIC:: ;;IMPLIC->PROP::) n ::PROP->,´Ñ:: ::MPL->ART::)::COPY ART-> ART->MOTtyp*) buf

Nota * significa input/output "externo", :: significa trasmisión sobre la red de datos; (…)buf significa procesamiento en buffer dentro de un subsistema, y (::….::)n significa posible procesamiento recíproco entre subsistemas.

Figura 3.18: Ejemplo de ICS

3.4.3.1.2. Descripción de la información empírica recogida con ánimo de sustentar su sistema experto.

Los desarrolladores de ICS muestran que existe cierta cantidad de conocimiento en HCI que puede ser utilizado para modelar tareas. Este conocimiento proviene de estudios realizados por los miembros del equipo por lo que

puede considerarse, por tanto, que se relaciona directamente con el sistema experto que intentan desarrollar. Consideraremos tres puntos: información acerca de la estructura de las tareas, acerca de objetos visuales y, por último, acerca de la disposición visual de los objetos en el interfaz.

Estructura de las tareas.

Determinar qué acción es la correcta en qué momento puede ser más o menos complicado según como se estructure la tarea a realizar. Para demostrar esto Barnard et al. (Barnard, et al., 1984) ofrecieron los mismos comandos a usuarios experimentales agrupándolos según dos estructuras jerárquicas distintas. En la primera de las estructuras existían constricciones lógicas en la ordenación de las tareas a un nivel intermedio, pero no a un nivel inferior. En la otra, las constricciones lógicas se daban al nivel inferior, pero no al intermedio. Los usuarios llevaron a cabo errores atendiendo a este patrón.

Este tipo de errores cometidos basándose en la estructura lógica de las tareas y no en el contenido semántico de éstas permite la modelización del sistema atendiendo a este aspecto. Para ello el sistema experto solicitará los niveles en los que el lenguaje de input se organiza, para, a continuación, preguntar si existen constricciones lógicas en la forma en que las acciones deben especificarse y diagnosticar un posible problema en caso de que la respuesta sea negativa.

Forma de objetos visuales.

May et al. (May, et al., 1993) proporcionan un modelo para el análisis de la ejecución en búsquedas basadas visualmente. Según ellos, los usuarios comparan la estructura visual de un objeto con una representación proposicional del objetivo generada internamente. Esta representación proposicional depende del contexto de otros elementos dentro del cual ha sido aprendida y/o ha realizado búsquedas. Por ejemplo, en los tres primeros iconos, la representación proposicional necesaria para llevar a cabo una búsqueda visual del segundo icono sería una representación que incluyera bien el atributo "rectángulo pequeño" o bien "triángulo" dado que los otros dos iconos no tienen ninguno de esos atributos. No obstante, en el segundo grupo de iconos el atributo que sería necesario es "triángulo a la derecha" ya que el resto de las características son semejantes a las del tercer icono. Asumiendo que los sujetos no son capaces de centrar su atención al primer vistazo en el atributo más significativo, es posible suponer que aquellos iconos que compartan más

características con otros iconos necesitarán un análisis más profundo que aquellos cuyas características sean más individuales.

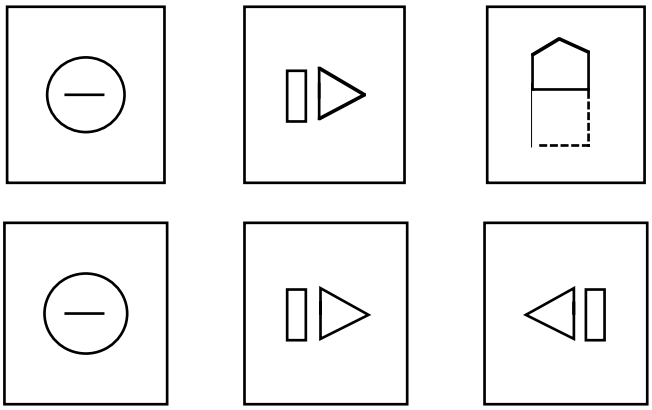


Figura 3.19: Iconos abstractos utilizados por Wandmacher et. al.

Esta afirmación es sustentada por la reinterpretación de un experimento llevado a cabo por (Wandmacher y Arend, 1985) y replicado por Valero et. al (Valero, et al., En preparación). En este experimento se presentaron dos conjuntos de iconos denominados abstractos y representativos a un grupo de sujetos. El resultado principal, que los iconos abstractos tenían tiempos de respuesta menores que los iconos representativos como puede verse en la figura 3.20, pero no es de importancia aquí. Lo importante es que, aplicando las ideas anteriormente señaladas, May et. al (May, et al., 1992) llevan a cabo una clasificación de los iconos en función del número de otros iconos con atributos similares que existen en cada grupo, ordenándolos por la profundidad de procesamiento que necesitan para ser identificado. Esta clasificación como es posible ver sigue una linea aproximadamente recta tanto en el estudio de Wandmacher et al como en el de Valero et. al. Debido a que la replicación es sólo parcial y no se tuvo en cuenta la condición que supuestamente debería haber generado diferencias más grandes (sólo cinco distractores frente a cinco y once distractores en el estudio original) resulta aceptable que la pendiente sea menor en ésta. En (May, et al., 1993) se proporciona una prueba más directa de las ideas sugeridas.

Disposición visual del interfaz.

Myers et. al (Myers y Hammond, 1991) al analizar un sistema de hipertexto descubrieron que eliminar un botón de un interfaz significó romper una unidad visual que acarreó la incomprensión de una función básica en el sistema. Ello se debió a que una estructura funcional estable (cuatro comandos) que se encontraba reflejada en una estructura visual (cuatro botones formando un grupo), fue rota al

modificar la estructura visual (un botón intermedio fue eliminado, lo que llevó a que los usuarios percibieran los botones como formando dos grupos en lugar de uno) lo que llevó a una peor comprensión del sistema (el botón aislado fue pobremente utilizado y comprendido).

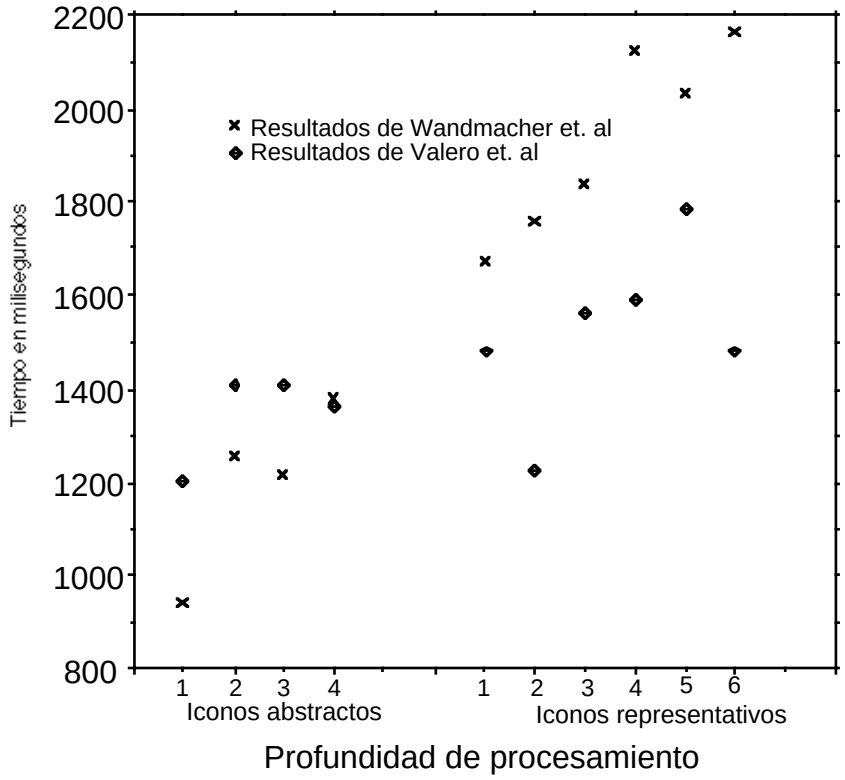


Figura 3.20: Tiempos medios de reconocimiento para iconos representativos y abstractos en dos estudios.

El sistema experto es capaz de preguntar acerca de relaciones funcionales y relaciones visuales, comprobando si existe un paralelismo entre ambas y dando diagnósticos negativos si eso no ocurre.

3.4.3.1.3. El sistema experto y su funcionamiento.

La complejidad del modelo de Barnard et. al. hace poco viable realizar análisis manuales. Por ello, han construido un sistema experto que se encarga de automatizar esos análisis y que libera al diseñador de esta tarea.

El sistema experto encargado de la modelización ha sido desarrollado por medio de un sistema comercial (Xi+ producido por Inference Ltd.), el cual ofrece

una arquitectura basado en reglas de conocimiento[9]. Estas reglas se agrupan en bases separadas, encargándose el conjunto de reglas de una base concreta de llevar a cabo determinadas tareas del sistema experto. En la figura 3.21 es posible ver el esquema de las bases de conocimiento del sistema experto (tomado de May, et al., 1992), así como los puntos en que el sistema va tomando información del encargado de hacer el análisis. Algunas de las bases de conocimiento más importantes, puestas con respecto al orden en que actúan son:

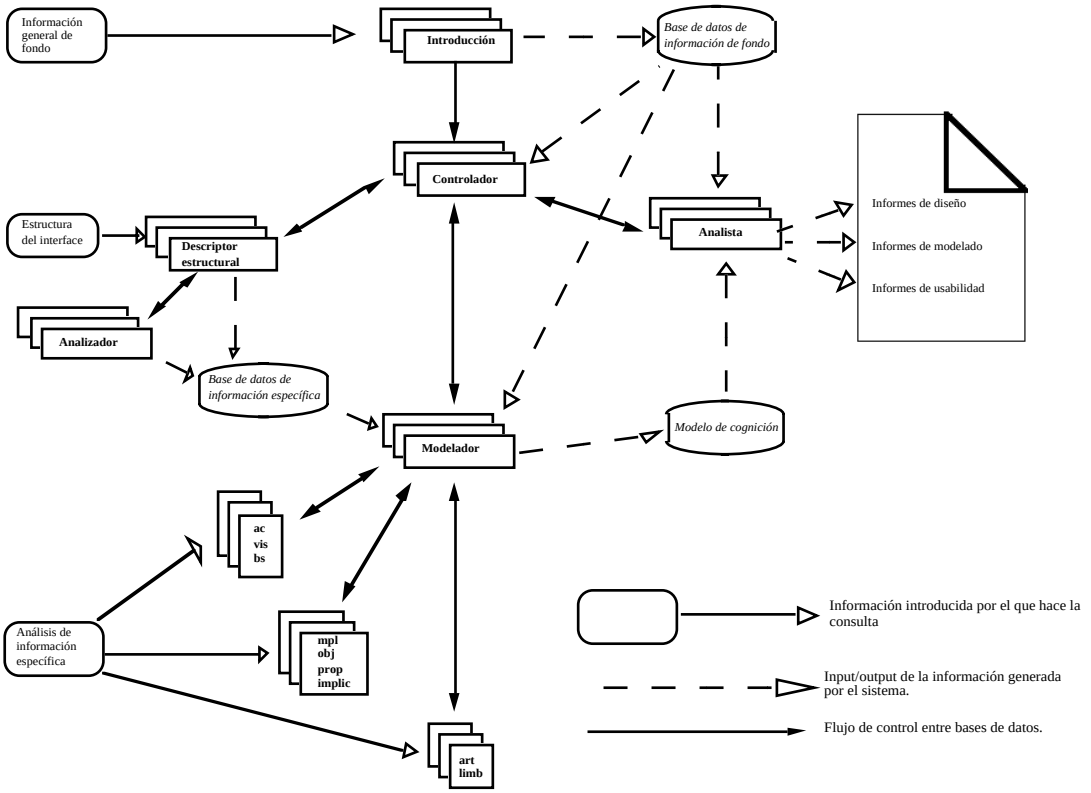


Figura 3.21: Una descripción esquemática de ICS Expert, el sistema experto para realizar análisis ICS.

1) El descriptor estructural: Esta base se encarga de la descripción estructural del interface. En ella, la información se divide atendiendo a las fases de formación de objetivos, especificación de la acción y ejecución de la acción (esta división se inspira en las ideas de Norman, tal y como son expuestas en p.e. Norman, 1986). Así, en primer lugar, el sistema pide información relevante con

[9] Aunque no está claramente especificado, las descripciones del sistema sugieren que este sistema básicamente ofrece una arquitectura de pizarra basada en reglas de producción (Hayes-Roth y Jacobstein, 1994).

respecto a la elaboración de los objetivos tal y como los pasos que la tarea necesitaría para ser llevada a cabo (por ejemplo, en un sistema de correo electrónico podría ser el advertir que ha sido recibido correo, el examen de las características del mensaje, la lectura de éste en su caso y su archivado o borrado). A continuación, el sistema solicita información acerca del ordenamiento que los pasos de la tarea tienen entre sí, dado que éste no posee ningún tipo de conocimiento semántico y es la persona haciendo el análisis la que le tiene que proporcionar la información que necesita acerca de estos pasos. Por ejemplo, en un sistema de correo electrónico existe un ordenamiento lógico entre el hecho de detectar que hay nuevo correo que se produciría en primer lugar y el hecho de examinar sus características que se produciría en segundo; otros sistemas no obstante no tienen porqué tener una ordenación similar.

El sistema pasaría a solicitar información acerca de las otras dos fases, especificación de la acción y ejecución de esta. Dado que muchos de los puntos en estas dos fases pueden considerarse como paralelos, el sistema ofrece una opción denominada Copiar para repetir la información introducida con cada elemento introducido.

2) El modelador: Esta base construye el modelo cognitivo atendiendo a la estructura de ICS. A su vez, esta base de conocimiento es capaz de llamar a otras, por lo que, por ejemplo, si el modelador revela que es necesario el pase de información desde el subsistema Objeto al subsistema Morfonoléxico, llamará a la base Obj, la cual incluye información acerca de este subsistema y puede por tanto determinar como de bien proceduralizada está la transformación, dada la naturaleza y estado de los códigos siendo trasformados. Esta base de datos puede asimismo solicitar información adicional acerca de puntos que este análisis puede anticipar que serán problemáticos. El resultado es una valoración en términos numéricos de la fase que está siendo analizada, lo cual servirá para que la siguiente base del conocimiento, el Analista, elabore sus informes.

3) El Analista: El analista lleva a cabo el tercer paso correspondiente al análisis, a partir del modelo de actividad cognitiva pasa a una descripción de la conducta del usuario o un comentario del diseño o una descripción del modelo en términos de ICS. Por ejemplo, si al llevar a cabo un análisis durante la especificación de la acción se otorga la calificación máxima positiva a la utilidad de las unidades básicas del sistema proposicional entonces se dispararía una regla

que indicaría que los usuarios ocasionales serían probablemente capaces de recordar este conjunto de comandos.

Una vez el informe ha sido producido, el modelador es capaz de volver atrás y, cambiando valores del diseño introducido repetir los análisis y obtener outputs distintos para de este modo evaluar las consecuencias de modificaciones en el diseño original.

Estudios empíricos con el sistema experto.

Shum y Hammond (1993) seleccionaron estudiantes de un curso de HCI y les introdujeron en el uso del sistema experto. Los sujetos llevaron a continuación una tarea que implicaba la evaluación de un sistema y su modificación por un alternativa. A continuación sigue una lista de los problemas encontrados a la hora de llevar a cabo esta tarea.

1) Saber qué fase se estaba modelando. ICS necesita información acerca de tres fases: Formación de objetivos, especificación de la acción y ejecución de la acción. Los sujetos que lo utilizaron no lograron captar las diferencias entre estos niveles y, más aún, tuvieron problemas en derivar las consecuencias relevantes en relación a ellos. Como resultado, tuvieron problemas en contestar adecuadamente a las preguntas que el sistema experto les iba haciendo.

2) Conocer la estructura de diálogo: Los sujetos eran conscientes en muchas ocasiones de en qué lugares era necesaria una mayor profundización en determinados aspectos, pero no sabían cuando se les iba a preguntar acerca de esos aspectos, por lo que se veían obligados a atender a cuestiones que ellos consideraban sin importancia antes de volver sobre el punto que realmente exigía, en su opinión, atención.

3) Conocer cómo definir grupos de comandos: Los grupos de comandos hacen referencia a la disposición visual de los comandos en el interface (ver apartado de disposición visual del interface). En un buen número de casos, los sujetos pensaron que la definición de grupos debería hacerse en función de la funcionalidad de los comandos, y no su presentación visual. Además, dependiendo de la fase de modelado, ciertas preguntas deben ser contestadas atendiendo a aspectos diferentes (p.e. las diferencias entre grupos de comandos pueden establecerse en términos de su funcionalidad, disposición en la pantalla y forma de ejecución), lo cual también provocó confusiones entre los sujetos.

4) Los usuarios dicen: "Estamos haciendo el trabajo de la herramienta": En varias ocasiones los usuarios manifestaron que tenían la impresión de estar llevando a cabo el trabajo de la herramienta, debido a que ciertas preguntas solicitaban el juicio del evaluador acerca de cuestiones que éste esperaba encontrar resueltas a través del sistema experto. Por ejemplo, el sistema experto pregunta cómo el significado de un comando encaja con el grupo en el que se halla incluido. Los usuarios sintieron que era responsabilidad del sistema experto contestar a eso, y que no era correcto que lo preguntara.

5) Los usuarios dicen: "Yo podía haber dicho eso sin necesidad de utilizar el sistema experto". Esta reacción parece bastante general. Hasta la fecha no hay ninguna prueba empírica que afronte esta cuestión.

3.4.3.1.4. CTA: Cognitive Task Analysis. análisis ICS manual.

Debido a que el sistema experto para los análisis ICS se encuentra todavía en una fase no adecuada para su uso general (aunque empezaron hace seis años), los desarrolladores de ICS utilizan en ciertas ocasiones análisis manuales que denominan CTA. El proceso no obstante no ha sido todavía explicitado en ningún documento que hasta la fecha haya sido publicado por lo que resulta difícil llevar a cabo ningún tipo de valoración de éste.

3.4.3.2. PUM (Programable User Models) de Young et. al.

El análisis basado en PUM parte de la descripción por un diseñador de interfaces del procedimiento deseado para llevar a cabo una tarea en un sistema interactivo, a partir del cual se deriva el conocimiento que sería necesario y/o estaría disponible para un usuario intentando llevarla a cabo. Ese conocimiento es codificado dentro de la arquitectura de espacios de problemas SOAR (Laird, et al., 1987a) produciendo un "modelo programado del usuario" que intentará llevar a cabo esta tarea. Este modelo proporciona información tanto acerca de las situaciones en las que el modelo necesitará embarcarse en actividades de solución de problemas o simplemente caerá en situación de "impass", requiriendo información externa para poder continuar llevando a cabo la tarea.

3.4.3.2.1. Antecedentes.

El principal antecedente de PUM es SOAR, un sistema de inteligencia artificial que intenta proporcionar una organización apropiada para la acción

inteligente. Este sistema está conectado con una línea de investigación que arranca con los trabajos de Newell y Simon acerca del "teórico lógico", la teoría general de solución de problemas y el desarrollo de sistemas de producción entre otros. A continuación ofreceremos unas notas acerca de SOAR tomados de los siguientes orígenes (Laird, et al., 1987b; Laird, et al., 1993) antes de continuar con la exposición de PUM.

En SOAR la tarea de lograr un objetivo es formulada en términos de encontrar un estado deseado en un espacio de problemas. Así, todas las tareas son resueltas por medio de búsquedas heurísticas. La conducta rutinaria surge cuando se dispone de suficiente conocimiento para proporcionar control de la búsqueda, es decir, para aplicar el operador correcto en cada situación.

Cuando una situación es problemática y no existe conocimiento para controlar la búsqueda, el sistema es capaz de generar un subobjetivo encargado de darle solución. A su vez este subobjetivo es capaz de llamar a otros subobjetivos. La situación problemática es denominada "impass".

Existe un tipo de conocimiento especial dirigido a controlar las búsquedas expresado en términos de preferencias.

El conocimiento a largo plazo es representado uniformemente por medio de un sistema de producción. Las producciones generan conocimiento de control de la búsqueda, conocimiento procedural para operadores simples (los operadores más complejos son realizados operando en una implementación del espacio de problemas) y realizan otras funciones.

Los métodos de solución de problemas básicos identificados en inteligencia artificial surgen directamente del conocimiento de la tarea. SOAR no necesita implementaciones especiales y separadas de estos métodos. Es capaz de generarlos por sí mismo.

En cada ocasión que el sistema genera un subobjetivo en un "impass", el cual llega a una solución satisfactoria, SOAR aprende la producción correspondiente de tal modo que en situaciones similares en el futuro no necesite llevar a cabo este proceso. Esto se denomina aprendizaje continuo por medio de "chunking".

Por último, comentar que no es un sistema experto intensivo en conocimiento en el cual haya mucho reconocimiento directo de situaciones problemáticas

implementadas desde un principio. SOAR deduce el comportamiento correcto en cada ocasión y va aprendiendo aquellos caminos que se revelan como adecuados. En las pruebas realizadas por los autores se sorprenden al descubrir que su arquitectura no necesita de un número excesivo de producciones añadidas para resolver problemas típicos de sistemas expertos con gran número de reglas.

3.4.3.2.2. Comparación con otros métodos de modelización del usuario.

Young et al. (Young, et al., 1989) señalan que el estilo de PUM es semejante al modelo de Kieras y Polson (1985) en ciertos aspectos. No obstante, existe un punto central que los distingue, ya que en un PUM no es necesario que estén especificados los métodos para llevar a cabo las tareas. Por el contrario, el objetivo de un PUM es observar si el modelo es capaz de generar la secuencia de acciones (tanto cognitivas como de otro tipo) necesarias para llevar a cabo la tarea. Además, su énfasis está menos en el análisis de la conducta proceduralizada y más en la consideración de conductas intensivas en el conocimiento (es decir, solución de problemas).

Howes y Young (Howes y Young, 1991) analizan el componente de conexión acción-tarea de un PUM apoyándose sobre trabajos previos con TAG. TAG también es capaz de llevar a cabo una modelización de la conducta del usuario de sistemas de ordenador, no obstante, posee dos debilidades que son analizadas en este trabajo: En primer lugar, escribir un TAG es como escribir un programa de ordenador, pero sin la ventaja de tener un interpretador para chequear que el programa es correcto. En segundo lugar, a pesar de las mejoras sugeridas por Howes y Payne (Howes y Payne, 1990) TAG falla en reflejar muchos aspectos del papel del dispositivo en guiar la conducta del usuario. No obstante, TAG es una metodología comprobada empíricamente y de hecho, los análisis de tipo racional de PUM coinciden en parte con las predicciones realizadas por este modelo, en el sentido de que interfaces consistentes en términos de TAG son aprendidos más fácilmente por un modelo de usuario expresado en términos de un PUM.

Por último, aunque PUM se apoya en una herramienta de inteligencia artificial (SOAR) la cual tiene conexiones evidentes con sistemas expertos (aunque su enfoque no intensivo en conocimiento marca diferencias incluso con estos), PUM no es un sistema experto en HCI tal y como sí es ICS, ya que lo modelado por PUM es un usuario, no un diseñador. De este modo, el conocimiento incorporado por los modelos PUM es el que supuestamente tendría un usuario interesado en

aprender a manejar el dispositivo, no el que tendría un diseñador decidiendo si el proyecto está o no bien planteado.

3.4.3.2.3. Evolución y estado actual.

PUM es una aproximación que conecta de una manera más o menos clara con intentos previos de modelización de la conducta del usuario. Así, Young et. al (1989) comentan que existe una cierta semejanza entre el estilo de la teoría de la complejidad cognitiva y su propuesta, y en (Howes y Young, 1991) se hace una comparación explícita entre TAG y PUM. Una diferencia más radical es la impuesta por la propia estructura de SOAR, el cual se centra en conducta menos proceduralizada y más relacionada con el conocimiento que los otros modelos. La otra gran diferencia estriba en el hecho de que los modelos anteriores debían ser analizados de un modo manual, siendo en buena parte un resultado de la actividad del diseñador el obtener una estimación de los posibles problemas mientras que SOAR ofrece un método para detectar automáticamente dónde podrían aparecer los problemas.

Actualmente, (Blandford y Young, 1993) las actividades en torno a PUM han consistido en desarrollar un lenguaje de instrucción (IL, Instruction Language) que permite llevar a cabo descripciones del conocimiento necesario acerca de la tarea y el dispositivo. Las descripciones llevadas a cabo en ese lenguaje serían analizadas por un interpretador denominado STILE que daría lugar a código SOAR, el cual sería un modelo ejecutable. No obstante, según señala Young (comunicación personal, 1994), ninguna de estas herramientas son hoy por hoy fáciles de usar en sí mismas así que su utilidad en un medio de desarrollo real está todavía por determinar.

En nuestro caso, hemos examinado el interpretador STILE con objeto de utilizarlo para realizar análisis a través de él. A pesar de nuestros esfuerzos y de haber superado algunas de las dificultades (STILE está escrito en lisp y genera código que luego es analizado por medio de SOAR) STILE no pudo ser utilizado de una manera mínimamente útil. La falta de documentación (no existe en absoluto), su carácter de herramienta de investigación, la falta de ejemplos y en general su estado inconcluso no nos permitieron realizar ningún tipo de análisis. Un problema que, como comentaremos más adelante, parece menor, ya que ni siquiera los autores utilizan estas herramientas para hacer sus análisis.

3.4.3.2.4. Investigaciones en torno a PUM.

No existe todavía lo que podríamos considerar pruebas empíricas del funcionamiento de PUM. Hasta ahora los análisis publicados han sido de tipo racional, comparando su funcionamiento potencial con resultados publicados en la literatura. Por ejemplo, (Young, et al., 1990) partiendo del conocimiento (supuesto) que tres tipos de usuarios (novato, intermedio y experto) tienen del interface de usuario de los ordenadores Macintosh, se plantean qué conocimiento necesitan tener o les es propio al elegir un archivo para abrirlo. Ello les lleva a una descripción de ese conocimiento que podría ser simulada en SOAR. Por otro lado, en (Young y Whittington, 1990) introducen una notación (aunque no definitiva) para llevar a cabo las descripciones.

Capítulo 4

Descripción y análisis de un interfaz hombre-computador

4.1 Introducción

En este capítulo se presenta una aplicación de tres modelos formales para la descripción de un ejemplo preparado a tal efecto. Estas descripciones permiten realizar una serie de predicciones acerca de diversas variables dependientes (tiempo de realización, errores, etc.) en relación con las tareas.

Pasaremos en primer lugar a justificar los modelos utilizados a la hora de describir la situación. En segundo lugar, veremos una descripción de la situación por la que los sujetos pasaron y las descripciones formales realizadas de esa situación. Por último, comprobaremos el ajuste de las predicciones a las descripciones realizadas.

4.2 Justificación de los modelos utilizados

Dentro de los objetivos de la tesis se contemplaba el realizar un diagnóstico de la aplicabilidad de las técnicas y la existencia de información suficiente para llevarlas a cabo con la suficiente fiabilidad. En este apartado realizaremos un adelanto de las conclusiones extraidas al respecto para de este modo dar continuidad al texto sin perjuicio de retomar algunas de las cuestiones tratadas aquí en el apartado de conclusiones.

En el capítulo anterior hemos visto una descripción de diversas aproximaciones a la descripción formal de interfaces hombre ordenador presentes en la literatura. Como ha quedado establecido, todas estas descripciones pueden ser clasificadas en tres apartados: descripciones formales del sistema, notaciones de descripción de diálogos y modelos de usuario.

Ya que nuestro interés se centra fundamentalmente en la predicción de las dificultades que los usuarios encontrarán a la hora de realizar tareas con el ordenador, los modelos de usuario son la perspectiva en que nos hemos centrado.

Con respecto al resto de las notaciones, las descripciones formales del sistema presentan una perspectiva excesivamente abstracta respecto al comportamiento real que un interface debería tener, estando más centrada en el establecimiento de los marcos conceptuales dentro de los que un diseño debería encontrarse antes que en la especificación de conductas concretas. Otro tipo de notaciones más concretas son necesarias para la especificación de sistemas concretos (Dix, 1991).

Las notaciones para la descripción de diálogos están un paso más cerca de la descripción de sistemas concretos. Por medio de ellas un diseñador puede establecer las acciones concretas que el usuario tendrá disponibles, los resultados de esas acciones y en general el comportamiento del interfaz. Los posibles beneficios de esta descripción son los derivados de establecer un diseño que pueda ser discutido en equipo, ser visualizado más claramente y ser reestructurado sin el excesivo esfuerzo que significaría hacer lo mismo con un prototipo funcional. También, y no menos importante en ocasiones,

podría ser acordado un contrato en función de estas especificaciones, evitando posteriores reclamaciones y desacuerdos.

No obstante, estas descripciones no fueron diseñadas con la perspectiva de realizar análisis y predicciones de la conducta del usuario, por lo que determinar los puntos conflictivos o débiles de una especificación es una tarea que recae sobre el diseñador, a partir de su propia experiencia. Por ello, su interés desde nuestro punto de vista puede considerarse como limitado[1].

Ahora bien, una de las notaciones de descripción de diálogos sí incorpora una serie de sugerencias para el análisis de las especificaciones. Esta notación es la denominada Action Task (Curry y Monk, 1990a; Curry y Monk, 1990b; Monk, 1990). Además, y como una aportación original de este trabajo, ha sido desarrollada una métrica basada en el número posible de acciones en cada momento (una explicación más detallada aparecerá más adelante) que permite una predicción de la dificultad de realización de cada tarea.

Por último, las dos siguientes notaciones para la descripción de modelos de usuario han sido utilizadas. Estas son, NGOMS, la versión más "amigable" de GOMS (Kieras, 1988; Kieras, 1991) y TAG (Schiele y Green, 1990). Dos razones fundamentales nos han llevado a esta elección:

1) El ser considerados como dos métodos ampliamente citados y de gran impacto en la literatura si atendemos a las revisiones existentes acerca del campo (p.e. Dix, et al., 1993; de Haan, et al., 1991; Simon, 1988).

2) El tener textos que especifican de una manera lo suficientemente clara y detallada la aplicación de los métodos de tal modo que su aplicación puede ser considerada como relativamente clara y fiable. Este tipo de explicaciones tan precisas no abundan precisamente en la literatura, estando en muchos casos muy limitada la aplicación de estos modelos por esta causa.

Otros métodos presentes en la literatura no cumplen estas características. Sin entrar en todos los métodos uno por uno resulta

1 En realidad, debido a la progresiva aparición de sistemas de prototipos de interfaces gráficos, esta perspectiva puede derivar en sistemas que permitan la especificación de prototipos funcionales de una manera relativamente sencilla y rápida.

interesante comentar las razones por las que se ha rechazado su utilización en dos casos muy concretos que presentan un volumen importante de apariciones en la literatura y un gran interés para muchos de los autores que realizan revisiones. Estos son PUM (Young, et al., 1989; Young y Whittington, 1990) e ICS (Barnard, et al., 1987; Barnard, 1988; May, et al., 1993).

La aplicación del primer método se encuentra supeditada en PUM a la utilización de una serie de rutinas escritas en lisp (denominadas STILE) que generan un código utilizable en un sistema experto (SOAR). Realizar esta descripción para luego proporcionársela a las rutinas en lisp es, al parecer, tan complejo, que, en la práctica, los propios autores realizan análisis verbales, aunque "inspirados" en las posibles descripciones que podrían hacerse. Así, en Blandford y Young (1994), después de haber realizado un análisis de un interfaz se comenta: "... dados los resultados de un análisis del conocimiento simple (que iluminó un gran número de inconsistencias y cuestiones no resueltas acerca del diseño), no hemos identificado ningún área en la que hacer un análisis PUM completo (que incluiría el desarrollo de una descripción en lenguaje de instrucción del diseño y la implementación de un modelo funcional) fuera una posibilidad a considerar". Este planteamiento, no obstante, impide la utilización de su método a otros investigadores o diseñadores, ya que no hay ninguna especificación clara de la forma en que es realmente realizado. Dos párrafos más allá, los propios autores contestan a esta cuestión del siguiente modo: "¿En qué cantidad esto es un modelado de usuario que podría ser enseñado como una habilidad independiente a un diseñador, y cuánto es el resultado de la inteligencia original de individuos concretos?". La respuesta es que los autores "intuyen" que buena parte del proceso que realizan es independiente de los modeladores. No obstante, lo cierto es que el nivel al que su método está expuesto, no ofrece muchas oportunidades para poder realizar comprobaciones de sus aseveraciones. En nuestro caso, al utilizar las rutinas nos hemos encontrado con grandes dificultades, ya que no existen instrucciones ni información acerca de cómo utilizarlas. Además, no existen ejemplos completos desarrollados, sino partes muy triviales de situaciones muy concretas. Aunque, muy probablemente, nuestra incapacidad para hacer uso de estas rutinas pueda deberse a la falta de conocimientos en un área de la complejidad de SOAR y la inteligencia artificial, no creemos que un método que descansa sobre la posesión de estos

conocimientos tenga nunca aceptación generalizada. De algún modo, los autores deberían proporcionar un conjunto de normas y procedimientos que cualquiera pudiera seguir.

Por otro lado, ICS tampoco tiene ninguna exposición que muestre de una manera formal el proceso que se sigue para llevar a cabo una descripción (v.p.e. May, et al., 1992; May, et al., 1993). Lo más parecido a una especificación clara de los métodos a seguir es el desarrollo de un sistema experto para el modelado de interfaces que debía de servir como una "prueba de concepto", el cual todavía no se encuentra disponible para uso general, y que, en nuestra experiencia con una versión "sólo para investigadores", se comportó de una manera poco fiable. Una experiencia, por otra parte, similar a la experimentada por diseñadores "del mundo real" (Shum y Hammond, 1993).

4.3 El ejercicio a describir

4.3.1 Elementos del ejercicio

Para la realización del ejercicio de modelado que nos propusimos llevar a cabo se elaboró un programa informático que serviría como ejemplo a analizar. Este programa constaba de cinco módulos que simulaban una tarea semejante a las que podrían ejecutarse mediante un procesador de textos. Las tareas simuladas fueron las siguientes:

1) Un ejemplo de cambiar comentario. 2) Un ejemplo de guardar archivo.

Figura 4.1: Pantallas que los usuarios vieron en su interacción con el ordenador.

1) Cambiar la primera línea de un texto.
2) Cambiar el comentario adherido a un texto.
3) Guardar el archivo utilizado con un nombre distinto.

4) Imprimir un texto.

5) Cambiar el tipo y el tamaño de las letras de un texto.

Algunas de las pantallas características que los usuarios utilizaron pueden verse en la figura 4.1.

El programa fue desarrollado de tal modo que los usuarios sólo podían seguir un camino hasta alcanzar el objetivo. De ese modo, los errores cometidos no llevarían a los usuarios a situaciones alejadas del problema a resolver, y la recuperación de error necesitaría la menor cantidad de tiempo posible. En caso de no realizar la acción correcta, el sistema respondía no llevando a cabo ninguna acción la primera vez. Cuando el sujeto realizaba la acción incorrecta de nuevo, el sistema respondía con un mensaje indicando al usuario cual era la acción correcta que debía seguir a continuación.

Debido a que se esperaba que muchos de los sujetos utilizados tuvieran una experiencia baja o nula en el uso de ordenadores, lo cual incluía problemas en la utilización de materiales de input como el ratón y el teclado, se introdujeron dos ejercicios de entrenamiento y aprendizaje inicial. Estos fueron, en primer lugar, un tutorial diseñado para aprender la utilización del ratón proporcionado por los fabricantes del ordenador utilizado, y, en segundo lugar, un tutorial desarrollado ad hoc para la enseñanza de elementos básicos de manejo del interfaz de usuario tales como botones, cuadros de diálogo, menús, etc. Un ejemplo del sistema utilizado para el aprendizaje de esos elementos básicos se muestra en la figura 4.2.

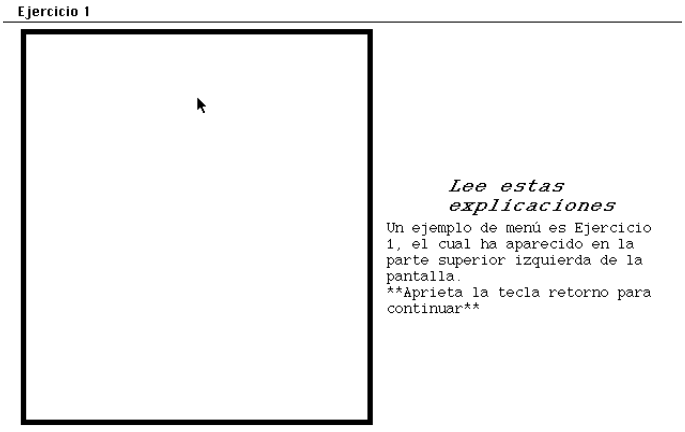


Figura 4.2: Sistema de enseñanza.

Por último, para la realización de cada tarea se proporcionó a los usuarios un material escrito que detallaba los pasos a seguir, así como una cierta

cantidad de información general acerca del significado y contenido de ésta. Estos textos pueden ser consultados en el Anexo I y sólo se permitía utilizarlos al principio de cada tarea, no durante su realización. De este modo, los sujetos estaban, en buena parte, obligados a aprender los procedimientos a partir de la información proporcionada en la pantalla.

4.3.2 Desarrollo de los ensayos

El desarrollo del ejercicio de manejo del programa descrito seguía los siguientes pasos.

1) Fase de aprendizaje: los sujetos pasaban por los dos tutoriales anteriormente descritos.

2) Una descripción del cuestionario NASA-TLX, destinado a medir carga mental, les era proporcionada a los sujetos. Después realizaban un ensayo para el manejo de este programa. Los resultados del pase de este cuestionario no serán mostrados aquí.

3) Los sujetos recibían instrucciones acerca de la primera de las tareas, retirándoselas después que estos manifestarán haberlas leído dos veces.

4) Los sujetos ejecutaban la tarea un mínimo de 3 veces de las cuales una de ellas al menos debería de haberse terminado sin cometer errores. En caso de que el usuario siguiera cometiendo errores ostensibles después del tercer ensayo se le hacía repetir la tarea hasta que éstos desaparecieran. Los sujetos fueron instruidos en realizar las tareas lo más rápidamente posible pero sin cometer errores. Cada uno de los intentos en las distintas tareas fueron analizados por separado lo cual conllevó una serie de complejidades de análisis que serán explicadas más adelante. Las tareas fueron realizadas en dos ordenes diferentes para de ese modo tener en cuenta los efectos de haber aprendido las tareas siguiendo series diferentes.

4.3.3 Método de registro de las interacciones

El desarrollo de las sesiones fueron registradas por medio de ScreenRecorder™, un programa que proporcionaba archivos en disco duro que contienen las llamadas a las rutinas gráficas en ROM que realizan los ordenadores Macintosh para dibujar en pantalla. Posteriormente, el programa Mediatracks™ es capaz de reproducir esa película, permitiendo

introducir cortes en la película y proporcionando estimaciones de la duración de los cortes con una precisión que llega hasta la décima de segundo. El programa permite retroceder y avanzar paso a paso la imagen por lo que los cortes realizados puede ser precisados a un nivel apropiado.

Es necesario tener en cuenta que los análisis posteriores fueron realizados a un nivel de agregación mayor que el utilizado al llevar a cabo el registro observacional, por lo que es de esperar que las imprecisiones en la observación deberían de tender a ser anuladas. Como es posible ver, un total de 3362 observaciones fueron obtenidos de las cintas individuales que describían el tiempo empleado por el sujeto en cada una de las tareas. Este volumen de datos implica naturalmente un gran esfuerzo de recogida. Como comentan (Ritter y Larkin, 1994) no es de extrañar la poca cantidad de estudios que utilizan este tipo de datos. Tampoco es de extrañar el número de sujetos tan reducido comparado con estudios basados en otros métodos.

	Media	Mediana	N	Asime.	Curto.	Dv.tí	Min	Max
Tiempo	2.456	1.200	3362	5.963	62.51	3.885	0	68.70
Error	24.12	9	3362	1.872	2.468	30.46	1.900	127

Tabla 4.1: Estadísticos descriptivos de los datos originales.

4.3.4 Muestra

Los sujetos fueron trece alumnos de primer curso de Estadística de la Facultad. El primero de ellos fue desechado debido a ser considerado como prueba para el manejo de los instrumentos y el software utilizados, por lo que los resultados que se mostrarán a continuación no tendrán en cuenta más que a doce usuarios.

4.4 Descripción de los modelos y predicciones cualitativas

Los modelos considerados para describir el sistema son NGOMS, TAG y Action-Task. Seguiremos este orden para mostrar las descripciones realizadas

4.4.1 Descripción NGOMS

La descripción NGOMS es una simplificación de CCT, el método propuesto por Bovair et al. (Bovair, et al., 1990) y que fue expuesta en su manera más clara en Kieras (1991).

Esta descripción presenta las siguientes estructura y características notacionales.

Estructura de la descripción

1) **Jerarquía:** La notación presenta una estructura jerárquica en la que los métodos más generales incluyen llamadas a métodos particulares. Cada uno de los niveles está etiquetado consecutivamente para permitir seguir esta estructura.

El primer nivel es especial ya que su única función es disparar los otros métodos que serán utilizados en cada caso por medio de una regla de selección que determina cual corresponde en cada caso.

Todos los métodos terminan con una regla que indica que el objetivo ha sido realizado, indicando que el control puede retornar a un nivel superior, tomándose la acción correspondiente.

2) **Profundidad de la jerarquía:** Kieras (1991) recomienda cuatro niveles aproximadamente de jerarquía. En el nivel inferior, las acciones ocurren al nivel de tarea más simple contemplado.

3) **Tareas definidas por el operador:** En el apartado denominado documentación es posible ver una descripción de las tareas definidas para la descripción. En ellas se encuentran por un lado los operadores externos primitivos y en segundo los operadores mentales. Estas entidades se definen del siguiente modo (Kieras, 1991):

• Operadores externos primitivos: Los operadores primitivos son operadores que no pueden ser refinados a un grano más reducido. Los operadores externos son acciones observables mediante las cuales los usuarios intercambian información con el exterior. El operador define los operadores externos primitivos en función del análisis a realizar.

• Operadores mentales: Son las acciones internas ejecutadas por los usuarios. Son no-observables e hipotéticas, inferidas por el teórico o el analista. Los operadores mentales hacen referencia a parámetros generales. Por ejemplo, el operador mental Verificar resultado <resultado> haría referencia a un resultado cualquiera. En nuestro caso, para facilitar la

lectura, en cada momento se indicará explícitamente el valor correspondiente.

4) **Supuestos y juicios llevados a cabo por el analista:** A la hora de realizar las descripciones se establecen una serie de supuestos que delimitan cuestiones particulares que podrían ser sujetos a posteriores comprobaciones.

La descripción puede seguirse a continuación. Para más información puede consultarse el capítulo anterior. Acompañando la descripción hay una serie de notas a pié de página que muestran algunos de los razonamientos o intuiciones extraídos a partir del análisis, los cuales podrían ser utilizados para guiar el desarrollo de un sistema y son un ejemplo del tipo de información obtenible al realizar este tipo de descripciones.

1. Nivel I
   1. Método para realizar objetivo general
      1. Obtener siguiente tarea del texto.
      2. Decidir: Si no más tareas simples, entonces devolver objetivo realizado.
      3. Llevar a cabo la siguiente tarea simple.
      4. Ir a 1.

      Reglas de selección para el objetivo de ejecutar la tarea siguiente.
      Si la tarea es Cambio de Tipo-Tamaño, entonces llevar a cabo Cambio de Tipo-Tamaño.
      Si la tarea es Imprimir, entonces llevar a cabo Imprimir.
      Si la tarea es Primera línea, entonces llevar a cabo cambio Primera línea.
      Si la tarea es Duplicar, entonces llevar a cabo Duplicar.
      Si la tarea es Cambiar Comentario, entonces llevar a cabo Cambiar Comentario.
2. Nivel II
   1. Método para llevar a cabo Cambio de Tipo-Tamaño
      1. Abrir archivo
      2. Cambiar TipodeLetra
      3. Cambiar TamañodeLetra
      4. Salir
      5. Devolver Objetivo realizado.
   2. Método para Imprimir
      1. Abrir archivo
      2. Ajustar Página
      3. Imprimir
      4. Salir
      5. Devolver Objetivo realizado.
   3. Método para Primera Línea
      1. Abrir archivo
      2. Cambiar PrimeraLinea
      3. Salir
      4. Devolver Objetivo realizado.
   4. Método para Duplicar
      1. Abrir archivo
      2. Guardar Archivo con otro nombre

```
        3.      Salir
        4.      Devolver Objetivo realizado.
    5.    Método para Cambiar Comentario
        1.      Cambiar el comentario[2]
        2.      Salir
        3.      Devolver Objetivo realizado.
3. Nivel III
    1.    Abrir archivo
        1.      Retener que el comando es Abrir y Seleccionar comando
      menú <comando>
        2.      Verificar resultado <Ventana abrir abierta>
        3.      Retener que el ítem de lista es <nombre de fichero> y
      Seleccionar de lista <ítem de lista>
        4.      Retener que Tipo de botón es Confirmar y Presionar
      botón <tipo de botón>
        5.      Verificar resultado <documento abierto>.
        6.      Devolver Objetivo realizado.
    2.    Cambiar TipodeLetra
        1.      Retener que comando de menú es <nuevo tipo de letra>
      y Seleccionar comando menú <comando>
        2.      Verificar resultado <Tipo de letra cambiado>
        3.      Devolver Objetivo realizado
    3.    Cambiar TamañodeLetra
        1.      Retener que comando de menú es <nuevo tamaño de
      letra> y Seleccionar comando menú <comando>
        2.      Verificar resultado <Tamaño de letra cambiado>
        3.      Devolver Objetivo realizado
    4.    Salir
        1.      Retener que el comando es Salir y Seleccionar comando
      menú <comando>
        2.      Verificar resultado <Salir del programa>
        3.      Devolver Objetivo realizado
    5.    Ajustar Página
        1.      Retener que el comando es Ajustar Página y
      Seleccionar comando menú <comando>
        2.      Verificar resultado <Ventana ajustar página abierta>
        3.      Retener que Tipo de botón es Confirmar y Presionar
      botón <tipo de botón>
        4.      Verificar resultado <página ajustada?[3]>
        5.      Devolver Objetivo realizado
    6.    Imprimir[4]
        1.      Retener que el comando es Imprimir y Seleccionar
      comando menú <comando>
        2.      Verificar resultado <Ventana Imprimir abierta>
```

2 Es posible que realizar el análisis partiendo de una tarea con un nivel tan amplio no sea correcto. Al fin y al cabo, podría ser dividida en fragmentos que presentarían consistencia con otras tareas. Después de los análisis estadísticos, esta es una hipótesis muy razonable. No obstante, este análisis ha sido conservado tal y como fue hecho originalmente.

3En este caso, el paso de verificar resultado es un tanto artificial. Obviamente se produce una verificación de que las cosas han ido bien, por que si no fuera así, el sistema produciría un error, sin embargo, no existe un resultado explícito que pueda ser contrastado, por lo que es dudoso que los sujetos estén contrastando esto o no. Este paso se queda, pero es indicativo de una cierta inconsistencia y es un resultado interesante del análisis. Por otro lado el Macintosh es exactamente así, produciendo este comando la mayor parte de las veces un resultado un tanto vago. Quizás un feed-back más explícito fuera valioso.

4 En este caso sí que aparecía una ventana indicando que el documento estaba siendo impreso.

3. Retener que Tipo de botón es Confirmar y Presionar botón <tipo de botón>
4. Verificar resultado <Ventana de impresión>
5. Devolver Objetivo realizado

7. Cambiar PrimeraLinea[5]
1. Retener que Tipo de botón es Primera Línea y Presionar Botón <Tipo de botón>
2. Verificar resultado <Ventana primera línea abierta>
3. Teclear Texto
4. Cerrar Ventana
5. Devolver Objetivo realizado

8. Guardar Archivo con otro nombre
1. Retener que el comando es Guardar Como y Seleccionar comando menú <comando>
2. Teclear Texto
3. Retener que Tipo de botón es Confirmar y Presionar botón <tipo de botón>
4. Devolver Objetivo realizado

9. Cambiar el comentario
1. Retener que comando de menú es Examinar y Seleccionar comando menú <comando>
2. Verificar resultado <Ventana comentarios abierta>
3. Retener que el ítem de lista es Comentario a cambiar y Seleccionar de lista <ítem de lista>
4. Retener que Tipo de botón es Cambiar comentarios y Presionar botón <tipo de botón>
5. Verificar resultado <Ventana cambiar comentarios abierta>
6. Teclear Texto
7. Retener que Tipo de botón es Confirmar y Presionar botón <tipo de botón>
8. Retener que Tipo de botón es Hecho y Presionar botón <tipo de botón>
9. Devolver Objetivo realizado

4. Nivel IV

1. Seleccionar comando menú <comando>

Reglas de selección para Seleccionar comando menú

Si comando apropiado en instrucciones entonces Seleccionar comando menú 1

Si no comando apropiado en instrucciones entonces Seleccionar comando menú 2

Seleccionar comando menú 1

1. Mirar instrucciones <comando de menú> y Registrar <comando de menú>.
2. Recuperar <comando de menú> y Localizar en menú <posición comando apropiado menú>[6] y Registrar <posición en menú>.
3. Recuperar <posición en menú> y Descender <menú apropiado>.
4. Recuperar <comando de menú> y Verificar resultado <comando de menú seleccionado>.

---

5 Una cosa que puede ocurrir: el sujeto teclea el texto y lo olvida. Luego, al cerrar la ventana, esta no lo hace, por lo que tiene que a) recordar el item de nuevo o b) buscarlo de nuevo en la documentación.

6 Kieras y Elkerton (p. 31) asumen que los usuarios expertos son capaces de recordar la posición de los comandos en los menús. En nuestro caso (v. Howes, 1990 para una discusión de este punto) es más apropiado pensar que ni siquiera los usuarios expertos recuerdan exactamente la posición de los comandos en menús y por tanto necesitan siempre llevar a cabo un proceso de localización para utilizarlos.

```
    5.    Soltar Botón ratón <comando apropiado menú>.
    6.    Paso compuesto: Olvidar <comando apropiado menú>,
  Olvidar posición <%%> y Devolver Objetivo realizado.
Seleccionar comando menú 2
    1.    Recuperar <comando de menú> y Localizar en menú
  <posición comando apropiado menú> y Registrar <posición en
  menú>
    2.    Recuperar <posición en menú> y Descender por menú.
    3.    Recuperar <comando de menú> y Verificar resultado
  <comando de menú seleccionado>.
    4.    Soltar Botón ratón <comando de menú>.
    5.    Paso compuesto: Olvidar <comando de menú>, Olvidar
  <posición %%> y Devolver Objetivo realizado.
2.  Seleccionar de lista <ítem de lista>
    1.    Recuperar <ítem de lista> y Mirar instrucciones <ítem
  de lista> y Registrar <ítem de lista>
    2.    Recuperar <ítem de lista> y Localizar <ítem de lista>
  y Registrar <posición %%>
    3.    Recuperar <posición %%> y Mover flecha-ratón a
  <posición %%>
    4.    Clickar
    5.    Recuperar <ítem de lista> y Verificar resultado <ítem
  correcto seleccionado>.
    6.    Olvidar <ítem de lista> y Olvidar <posición %%> y
  Devolver Objetivo realizado.
3.  Presionar botón [7]
    1.    Recuperar <tipo de botón> y Localizar <tipo de botón>
  y Registrar <posición %%>
    2.    Recuperar <posición %%> y Mover flecha-ratón
  <posición %%>
    3.    Clickar
    4.    Verificar resultado <botón resaltado invertido>
    5.    Olvidar <posición %%> y Devolver Objetivo realizado
4.  Teclear Texto
    1.    Mirar instrucciones <texto apropiado> y Registrar
  <texto apropiado>
    2.    Recuperar <texto apropiado> y Usar el teclado
    3.    Verificar resultado <Texto bien escrito>[8]
    4.    Olvidar <texto apropiado> y Devolver Objetivo
  realizado
5.  Cerrar ventana [9]
```

7 Este punto podría resolverse de otra manera, podría haberse descrito un único procedimiento para presionar botones, pero luego haber introducido una regla de selección que distinguiera entre botónes no seleccionados y botones seleccionados. Esta forma de describirlo no obstante tiene cierto interés intuitivo. Los botones que no son para confirmar en general llevan algún tipo de carga de significado y por lo tanto implican algún tipo de decisión más compleja que los que sólo sirven para confirmar. Ello hace que los botones de confirmar sean contestados de una manera más rápida e irreflexiva.

Otra diferencia estaría en el uso de la memoria a corto plazo. Los botones por defecto no tendrían que ser recordados por sí mismos y por tanto no haría falta un proceso de identificación, reconocimiento como en los otros tipos de botones.

8 Este verificar resultado quizás debería considerarse como un poco dudoso. ¿Hasta qué punto es similar a otros verificar resultado? Esta verificación se hace paso a paso a la vez que escribes y puede ser bastante diferente a otros verificar resultado.

9 Podría considerarse que esto es un caso específico de botón. Sin embargo la diferencia en la verificación hace que lo haya considerado como algo separado.

```
   1.     Recuperar <botón cuadro de cierre>[10] y Localizar
<botón cuadro de cierre> y Registrar <posición %%>
   2.     Recuperar <posición %%> y Mover flecha-ratón
<posición %%>
   3.     Clickar
   4.     Verificar resultado <La ventana se cierra
apropiadamente>
   5.     Olvidar <posición %%> y Devolver Objetivo realizado
```

Después de la descripción es necesario añadir la documentación en la que se especifica información necesaria para entender la descripción anterior (v. explicaciones anteriores).

Operadores externos primitivos

1. Descender por menú: Con el menú bajado equivale a mover la flecha del ratón hacia abajo hasta ponerla sobre el comando apropiado.
2. Soltar Botón ratón.
3. Mover flecha-ratón.
4. Click-ar.
5. Usar el teclado.

Operadores mentales.

1. Mirar instrucciones (<comandos>, <datos>, <instrucción apropiada>, <texto apropiado>): Significa mirar a una página con parámetros acerca de las tareas realizadas habiendo sido previamente señalados los valores correspondientes a la tarea por el experimentador.

2. Verificar resultado (<comando menú seleccionado> <nombre de archivo seleccionado> <texto bien escrito> <ventana cerrándose apropiadamente).Significa comprobar que la acción llevada a cabo da lugar a un resultado apropiado. También podría formularse como comprobar que la acción llevada a cabo produce algún tipo de resultado

10 Esta regla no es muy clara. Se trata de que el botón apropiado puede estar en la memoria del usuario o no. Si no lo está el sujeto tiene que entrar en una cierta fase de solución de problemas hasta dar con el botón apropiado. Esta fase de solución de problemas, no obstante no es necesariamente negativa, ya que tras ella se detecta no sólo el botón apropiado, sino también la posición en la que está. Ello puede ayudar a explicar el fenómeno de que los usuarios de sistemas gráficos se apoyen de una manera tan grande en la pantalla y mucho menos en la LTM de lo que quizás podría esperarse, ya que la diferencia en longitud no es tan larga. Hay que tener en cuenta que este proceso incumple una de las normas de Kieras, pero también hay que tener en cuenta que es debido a una cierta peculiaridad que permite “conducta aventajada”.

visible que no es el mensaje de error con las orientaciones acerca del camino correcto siguiente.

3. Localizar (<nombre de archivo> <comentario> <botón cuadro de cierre> <tipo de botón> <ítem de lista>).Significa localizar en pantalla alguna de las entidades señaladas entre paréntesis para luego, típicamente, señalarlas con el ratón o alguna acción similar. Este tipo de localización necesita meramente una exploración visual así que debería ser relativamente rápida y sencilla.

4. Localizar en menú (<posición comando apropiado menú>): Localizar en menú es una operación más complicada que localizar en pantalla que involucra bajar un menú y explorar hasta localizar el comando y, en caso de ser necesario, volver a mirar en otro menú.

5. Descender el menú: Descender el menú implica seleccionar el título de menú y luego bajar la flecha del ratón sin soltar el botón hasta seleccionar el comando apropiado.

Supuestos y juicios llevados a cabo por el experimentador.

- Los sujetos tienen un conocimiento del lenguaje de especificación de comandos que va desde nulo hasta bajo.

- Verificar resultado se supone que es percibida similarmente por los usuarios a pesar de que a menudo corresponde a efectos totalmente distintos en la pantalla.

- Ajustar Página en realidad no ofrece una retroalimentación tan explícita como debería[11].

- Cuando el sistema responde con un mensaje indicando cual es el camino correcto, el usuario percibe que se ha equivocado y lee el mensaje.

11Este es probablemente uno de los hallazgos más interesantes del ejercicio de descripción llevado a cabo. Una sugerencia es que por ejemplo en los procesadores de texto debería haber una indicación explícita y siempre visible acerca de cual es la impresora seleccionada y que tipo de papel es el que se está utilizando (y en qué orientación).

• Los usuarios no saben al principio, y luego no recuerdan claramente, las posiciones de los comandos en los menús así que necesitan involucrarse en un proceso de localización que puede ser más o menos largo. Este proceso se materializa en un explorar menús hasta encontrar un ítem que se aproxima a lo buscado.

• Cerrar ventana no es considerado un botón normal, sino que es descrito por separado.

Análisis del modelo

Para el análisis del modelo, dos tipos distintos de información serán extraidas. En primer lugar, predicciones cuantitativas obtenidas a partir del número de reglas totales que hay en cada método además de las reglas nuevas (no aprendidas anteriormente). En segundo lugar, y tomando las sugerencias hechas por Kieras (1991), hemos extraído algunos puntos que podrían ser origen de dificultades, los cuales también serán evaluados empíricamente. Antes de dar los resultados mostraremos las reglas proporcionadas por Kieras (1991).

Reglas para realizar el recuento de reglas de producción.

Las reglas son las siguientes:

• La sentencia de comienzo de un método cuenta como una regla.

• Cada paso cuenta como una regla. Por ejemplo:

4. Salir

Ello no impide que cuando ese paso llama a un método que incluye otros pasos no deban ser contados igualmente. De este modo se produce una suma acumulada de valores en los niveles superiores de la jerarquía de acciones.

• La sentencia de regla de selección también cuenta como una regla.

• El Sí...Entonces usado en una regla de selección también cuenta como una regla.

• La regla final de una regla de selección también cuenta como una regla. Por ejemplo:

3. Devolver Objetivo realizado

Recuento de reglas de producción.

A continuación se muestran los resultados de los recuentos de las reglas de producción. Para una explicación más detallada de un ejemplo de la realización de este recuento dirigirse al Anexo II.

Nivel al que consideramos el recuento	Método de referencia	Número de reglas
Nivel II		
1.	Método para llevar a cabo Cambio de Tipo-Tamaño	70
2.	Método para Imprimir	86
3.	Método para Primera Línea	68
4.	Método para Duplicar	69
5.	Método para Cambiar Comentario	47[12]
Nivel III		
1.	Abrir archivo	28
2.	Cambiar TipodeLetra	12
3.	Cambiar TamañodeLetra	12
4.	Salir	12
5.	Ajustar Página	20
6.	Imprimir	20
7.	Cambiar PrimeraLinea	23
8.	Guardar Archivo con otro nombre	24
9.	Cambiar el comentario	31
Nivel IV[13]		
1.	Seleccionar comando menú.	8[14]
2.	Seleccionar de lista	7
3.	Presionar botón	6
4.	Teclear Texto	5
5.	Cerrar ventana	6

TABLA 4.2: Número de reglas de las tareas consideradas.

12 Debido a problemas con el software, dos tareas iniciales fueron eliminadas y no consideradas en los análisis.

13 Dado que las tareas a este nivel fueron entrenadas previamente se dieron por ya conocidas, por lo que no fueron incluidas en los recuentos y posteriores análisis.

14 Aunque en la descripción se consideran dos métodos para realizar esta subtarea en realidad pueden considerarse practicamente el mismo, aunque con una pequeña complejidad añadida debido a la existencia de una regla de selección. Debido a ello, se contará el número de pasos en el método más largo y se añadirá una regla más por la existencia de una regla de selección.

Reglas para considerar la consistencia.

Una de las cuestiones claves en NGOMS es su capacidad de captar el solapamiento de reglas, colapsando las reglas redundantes y eliminando el tiempo que supondría aprender a manejarlas en situaciones posteriores que reutilizan esas reglas. Más específicamente, el procedimiento para estimar la transferencia del conocimiento es como sigue (Kieras, 1991):

1. Encontrar candidatos para la transferencia. Para ello hay que identificar métodos que tengan objetivos similares aunque no idénticos. Se considerarán candidatos cuando bien el verbo o bien el objeto del método sean semejantes. Si ambos son diferentes no ha posibilidad de transferencia.

2. Generalizar objetivos de los métodos. Para ello es necesario cambiar aquellos valores similares a un parámetro en ambos métodos. Por ejemplo, en nuestro caso los métodos de Cambiar Primera línea y Cambiar Nombre Archivo son cambiados a Cambiar algo PrimeraLínea-NombreArchivo. PrimeraLínea-NombreArchivo es quizás demasiado farragoso pero tiene la ventaja de facilitar la lectura de las reglas atendiendo a la consistencia.

3. Contar sentencias similares. Si los dos métodos no son similares, comenzar desde el principio de estos dos métodos e ir hacia abajo a través de los pasos y contar el número de sentencias NGOMS que sean idénticas en ambos métodos. Si la única regla similar es la sentencia de Método no hay verdadera consistencia, pero si hay otras se añade este valor al recuento de la consistencia.

4. Descontar las sentencias similares del tiempo de aprendizaje. Las sentencias idénticas contadas de esta manera son las únicas clasificadas como similares. Siguiendo el modelo de transferencia propuesto en NGOMS, sólo la primera aparición de las sentencias requiere aprendizaje completo. Las sentencias encontradas posteriormente vienen "libres de carga". Se descuenta el número de sentencias similares del número total de sentencias a aprender.

Pasaremos en primer lugar a exponer una justificación de la consistencia a partir del colapsamiento de métodos que hemos considerado aceptables para luego ofrecer los valores correspondientes en términos de reglas nuevas.

Justificación de la consistencia.

La consistencia puede ser considerada a varios niveles de la jerarquía de acciones. Los niveles están indicados en la parte superior:

Nivel II

En este nivel, pueden realizarse dos colapsamientos. Primera línea y Duplicar comparten una estructura que implica abrir un archivo, realizar un cambio en una caja de texto y salir del programa. Cambiar Tipo-Tamaño-Imprimir combina también tareas con una estructura similar en la que es necesario abrir un archivo, realizar cambios en dos lugares y luego salir. Cambiar Comentario no puede ser colapsado dado que empieza por una secuencia inusual (Abrir comentarios en lugar de abrir archivo, algo que quizás podría ser una propuesta de diseño).

```
Método para realizar Primera Línea-Duplicar.
Abrir Archivo.
Cambiar algo Primeralínea-NombreArchivo
Salir.
Método para realizar Cambio Tipo-Tamaño-Imprimir.
Abrir Archivo.
Cambiar algo Tipo-AjustarPágina.
Cambiar algo Tamaño-Imprimir.
Salir.
```
[15]

Nivel III.

A este nivel tenemos en primer lugar el método para Cambiar-TipoTamaño-Salir que comparten el ser métodos que se componen únicamente de utilizar un comando de menú. En segundo lugar, tenemos Cambiar-Comentario-Abrir que parten de acciones parecidas para seleccionar un archivo sobre el que realizar una serie de operaciones, aunque

15 Cambiar Comentario no presenta consistencia con los otros métodos tal y como se ha realizado esta descripción por lo que no aparece en este apartado.

Cambiar-Comentario continua con otras operaciones que ponen un límite a la consistencia existente. Por último, Imprimir y Ajustar-página son métodos que se componen de seleccionar un comando de menú y apretar un botón de confirmación.

Método para Cambiar TipoTamaño[16]-Salir.
1. Retener que comando de menú es <Cambiartamaño, texto o salir> y Seleccionar comando menú <comando>
2. Verificar resultado <Tamaño de letra cambiado, tipo de letra o el programa se cierra>
3. Devolver Objetivo realizado

Método para Cambiar Comentario-Abrir
1. Retener que el comando es Abrir-Examinar y Seleccionar comando menú <comando>
2. Verificar resultado <Ventana abrir-cambiar comentario abierta>
3. Retener que el ítem de lista es <nombre de fichero-comentario> y Seleccionar de lista <ítem de lista>
4. Retener que Tipo de botón es Confirmar-CambiarComentario y Presionar botón <tipo de botón>
5. Verificar resultado <documento abierto-ventanacomentariosabierta>.
<FIN CONSISTENCIA>

Método para Ajustar Página-Imprimir
1. Retener que el comando es Ajustar Página-Imprimir y Seleccionar comando menú <comando>
2. Verificar resultado <Ventana ajustar página abierta-Imprimir>
3. Retener que Tipo de botón es Confirmar y Presionar botón <tipo de botón>
4. Verificar resultado <página ajustada-Impresión realizándose>
5. Devolver Objetivo realizado[17]

Recuento atendiendo a la consistencia.

Debido a que los sujetos recibieron las tareas en dos ordenaciones distintas, se muestran los resultados para ambas. Para un ejemplo de como realizar los recuentos, véase Anexo II.

16 No confundir este Cambiar Tipo Tamaño a nivel III con el correspondiente a nivel II que englobaría también el abrir el archivo, realizar el cambio y posteriormente el Salir.

17 En este caso ni Cambiar Primera línea ni Guardar Archivo con otro nombre han sido puestos en relación con otros métodos.

Nivel	Método de referencia	Número de reglas
	ORDEN 1	
Nivel II		
1.	Método para llevar a cabo Cambio de Tipo-Tamaño	70
2.	Método para Primera Línea	68
3.	Método para Cambiar Comentario	47
4.	Método para Duplicar	0
5.	Método para Imprimir	0
Nivel III		
	Cambio de Tipo-Tamaño	
1.	Abrir archivo	28
2.	Cambiar TipodeLetra	12
3.	Cambiar TamañodeLetra	0
4.	Salir	0
	Primera Línea	
5.	Abrir archivo	0
6.	Cambiar PrimeraLinea	23
7.	Salir	0
	Cambiar Comentario	
8.	Cambiar el comentario	15
9.	Salir	0
	Duplicar	
10.	Abrir archivo	0
11.	Guardar Archivo con otro nombre	24
12.	Salir	0
	Imprimir	
13.	Abrir archivo	0
14.	Ajustar Página	20
15.	Imprimir	0
16.	Salir	0
	ORDEN 2	
Nivel II		
1.	Método para llevar a cabo Cambio de Tipo-Tamaño	70
2.	Método para Imprimir	0
3.	Método para Primera Línea	68
4.	Método para Duplicar	0
5.	Método para Cambiar Comentario	47
Nivel III		
	Cambio de Tipo-Tamaño	
1.	Abrir archivo	28
2.	Cambiar TipodeLetra	12
3.	Cambiar TamañodeLetra	0
4.	Salir	0

	Imprimir	
5.	Abrir archivo	0
6.	Ajustar Página	20
7.	Imprimir	0
8.	Salir	0
	Primera Línea	
9.	Abrir archivo	0
10.	Cambiar PrimeraLinea	23
11.	Salir	0
	Duplicar	
12.	Abrir archivo	0
13.	Guardar Archivo con otro nombre	24
14.	Salir	0
	Cambiar Comentario	
15.	Cambiar el comentario	15
16.	Salir	0

Tabla 4.3: Valoración de la consistencia mediante NGOMS.

Tiempo de aprendizaje[18]

En Kieras (1991) se ofrece la siguiente fórmula para calcular el tiempo de aprendizaje.

Tiempo de aprendizaje=(30-60)minutos+30 segundos X número de sentencias NGOMS a aprender.

Así, se espera un valor inicial de entre 30 a 60 minutos para el sistema total y luego cada nueva regla de producción añadiría 30 segundos. En este caso mostraremos las estimaciones para cada uno de los métodos y dividiremos el tiempo inicial de modo constante entre todos los métodos.

- **Sin tener en cuenta las ganancias de la consistencia.**

	Nivel II	
1.	Cambio de Tipo-Tamaño	6-12+35 m
2.	Imprimir	6-12+43.5 m
3.	Primera Línea	6-12+34 m
4.	Duplicar	6-12+44.5 m
5.	Cambiar Comentario	6-12+23.5 m

18 De momento sólo he intentado hacer predicciones para el nivel II y el nivel III. El nivel IV no ha sido considerado.

	Nivel III	
1.	Abrir archivo	3.4-6.7+14 m.
2.	Cambiar TipodeLetra	3.4-6.7+6 m.
3.	Cambiar TamañodeLetra	3.4-6.7+6 m.
4.	Salir	3.4-6.7+6 m.
5.	Ajustar Página	3.4-6.7+10 m.
6.	Imprimir	3.4-6.7+10 m.
7.	Cambiar PrimeraLinea	3.4-6.7+11.5 m.
8.	Guardar Archivo con otro nombre	3.4-6.7+12 m.
9.	Cambiar el comentario	3.4-6.7+15.5 m.

- **Teniendo en cuenta las ganancias de la consistencia.**

Orden 1

	Nivel II	
1	Cambio de Tipo-Tamaño	6-12 + 35 m.
2	Primera Línea	6-12+34.5 m.
3	Cambiar Comentario	6-12+23.5 m.
4	Duplicar	6-12+0 m.
5	Imprimir	6-12+0 m.

	Nivel III	
1	Abrir archivo	3.4-6.7+14
2	Cambiar TipodeLetra	3.4-6.7+6
3	Cambiar Tamaño de Letra	3.4-6.7
4	Salir	3.4-6.7
5	Abrir archivo	0
6	Cambiar PrimeraLinea	3.4-6.7+11.5
7	Salir	0
8	Cambiar el comentario	3.4-6.7+12.5
9	Salir	0
10	Abrir archivo	0

11	Guardar Archivo con otro nombre	3.4-6.7+12
12	Salir	0
13	Abrir archivo	0
14	Ajustar Página	3.4-6.7+10
15	Imprimir	3.4-6.7
16	Salir	0

Orden 2

	Nivel II	
1	Cambio de Tipo-Tamaño	6-12 + 35 m.
2	Imprimir	6-12 + 0 m.
3	Primera Línea	6-12 + 34m.
4	Duplicar	6-12 + 0 m.
5	Cambiar Comentario	6-12 + 23.5 m.

	Nivel III	
1	Abrir archivo	3.4-6.7+14
2	Cambiar Tipo de Letra	3.4-6.7+6
3	Cambiar Tamaño de Letra	3.4-6.7
4	Salir	3.4-6.7
5	Abrir archivo	0
6	Ajustar Página	3.4-6.7+10
7	Imprimir	3.4-6.7
8	Salir	0
9	Abrir archivo	0
10	Cambiar PrimeraLinea	3.4-6.7+11.5
11	Salir	0
12	Abrir archivo	0
13	Guardar Archivo con otro nombre	3.4-6.7+12
14	Salir	0
15	Cambiar el comentario	3.4-6.7+15
16	Salir	0

Tiempo para aprender información en la memoria a largo plazo.

Para Kieras (1991) el tiempo para aprender de la memoria a largo plazo significa estimar el número de "chunks" que es necesario almacenar para llevar a cabo una tarea. Un número excesivo de "chunks" puede enlentecer el tiempo necesario para aprender un método y por tanto una cierta cantidad de tiempo debería ser añadida por cada ítem extra.

No necesitar recuperar muchos elementos de la memoria a largo plazo es precisamente una de las ventajas usualmente más mencionadas de los interfaces gráficos, ya que la información está disponible en la pantalla y los usuarios no necesitan recuperarla de la memoria para poder utilizarla. Por ejemplo, los usuarios exploraran un menú en busca de un ítem que se aproxime a la descripción aproximada de la tarea que se pretende realizar, tratándose por tanto de un proceso de reconocimiento[19], diferente del proceso de recuerdo.

En nuestro ejemplo, no hay ningún caso en el que los usuarios tuvieran que aprenderse y posteriormente recordar ítems en la memoria a largo plazo sin ningún tipo de ayuda externa, por lo que no se ha añadido ningún tipo de tiempo extra por esta causa.

Tiempo de ejecución.

Kieras (1991) ofrece la siguiente fórmula de cálculo del tiempo de ejecución.

Tiempo de ejecución = Tiempo por regla de producción NGOMS + Tiempo para los operadores externos primitivos +Tiempo de operadores mentales definidos por el analista + Tiempo de espera.

En donde:

• Tiempo por regla de producción NGOMS= 0.1 s.

• Tiempo de operadores externos primitivos. Usando los valores dados por Card, Moran y Newell (1983) tenemos:

19 Es curioso señalar que en el modelo de Kieras no se contempla el proceso de reconocimiento, el cual tiene unas características diferentes de los procesos de recuerdo o de uso de elementos en la memoria a corto plazo.

0.28 seg. por cada pulsación.

0.1 seg. por la presión del botón del ratón o su liberación.

1.1 seg. (como promedio) por un movimiento de ratón.

0.4 seg. al mover la mano al ratón o al teclado.

• Tiempo para operadores mentales definidos por el analista. Una estimación muy general puede ser 1.2 segundos por cada operador (Olson & Olson, 1989).

• Tiempo de espera: Es el tiempo que el usuario está esperando la contestación del ordenador. En nuestro caso este tiempo es negligible y en cualquier caso ha sido eliminado de los datos obtenidos en la observación.

Nivel I		Número de reglas	Operadores externos primitivos[20]	Tiempo operador mental definido por el analista [21]
1.	Método para llevar a cabo Cambio de Tipo-Tamaño	75	MR + SR + MR + CR + MR + SR + MR + SR + MR + SR	L + V + V + MI + L + L + V + V + L + V + V + L + V + V + L + V + V
2.	Método para Imprimir	109	MR + SR + MR + CR + MR + SR + MR + CR + MR + SR + MR + CR + MR + SR	L + V + V + MI + L + L + V + V + L + V + V + L + V + V + L + V + V + L + V + V + L + V + V
3.	Método para Primera Línea	79	MR + SR + MR + CR + MR + CR + 6*T + MR + CR + MR + SR	L + V + V + MI + L + L + V + V + L + V + V + L + V + L + V + V
4.	Método para Duplicar	86	MR + SR + MR + CR + MR + SR + MMT + 6*T + MR + CR + MR + SR	L + V + V + MI + L + L + V + V + LM + V + L + V + V
5.	Método para Cambiar Comentario	75	MR + SR + MR + CR + MR + CR + MMT + 6*T + MR + CR + MR + CR + MR + SR	LM+ V + V + MI + L + L + V + L + V + L + V + L + V + V
Nivel II				

20 Los siguientes códigos son utilizados: MR=Mover Ratón, SR: Soltar botón Ratón, CR: Clicar Ratón, MMT: Mover la Mano al Teclado, T: Pulsar una tecla o usar el teclado.

21 Han sido utilizados los siguientes códigos: L=Localizar, V=Verificar, MI=Mirar Instrucciones, LM=Localizar en menú.

1.	Abrir archivo	34	MR + SR + MR + CR	L + V + V + MI + L + L + V + Verificar
2.	Cambiar TipodeLetra	18	MR + SR	L + V + V
3.	Cambiar TamañodeLetra	18	MR + SR	L + V + V
4.	Salir	18	MR + SR	L + V + V
5.	Ajustar Página	26	MR + SR + MR + CR	L + V + V + L + V + V
6.	Imprimir	26	MR + SR + MR + CR	L + V + V + L + V + V
7.	Cambiar PrimeraLinea	23	MR + CR + 6*T + MR + CR	L + V + V + L + V
8.	Guardar Archivo con otro nombre	30	MR + SR + MMT + 6*T + MR + CR	LM + V
9.	Cambiar el comentario	54	MR + SR + MR + CR + MR + CR + MMT + 6*T + MR + CR + MR + CR	LM + V + V + MI + L + L + V + L + V + L + V
Nivel III				
1.	Seleccionar comando menú.	14	MR + SR	LM + V
2.	Seleccionar de lista	7	MR + CR	MI + L
3.	Presionar botón	6	MR + CR	L + V
4.	Teclear Texto	5	MMT + 6T[22]*	
5.	Cerrar ventana	6	MR + CR	L + V

La aplicación de las fórmulas produce los siguientes resultados:

Nivel I	Tiempo operadores primitivos (s)	Tiempo operadores mentales (s)	Tiempo reglas de producción (s)	Tiempo total (s)
Método para llevar a cabo Cambio de Tipo-Tamaño	6	20.4	7	33.4
Método para Imprimir	8.4	27.6	8.7	44.7
Método para Primera Línea	7.68	19.2	6.8	33.68
Método para Duplicar	8.08	15.6	8.6	32.28

22 Un valor intermedio de la longitud de los textos que tuvieron que teclear fue de aproximadamente 6 caracteres.

Método para Cambiar Comentario	9.28	16.8	6.9	32.98
Nivel II				
Abrir archivo	2.4	9.6	2.8	14.8
Cambiar TipodeLetra	1.2	3.6	1.2	6
Cambiar TamañodeLetra	1.2	3.6	1.2	6
Salir	1.2	3.6	1.2	6
Ajustar Página	2.4	7.2	2	11.6
Imprimir	2.4	7.2	2	11.6
Cambiar Primera Linea	4.08	6	2.3	12.38
Guardar Archivo con otro nombre	4.48	2.4	2.4	9.28
Cambiar el comentario	8.08	13.2	3.1	24.38

Carga mental.

Kieras (1991) ofrece una serie de indicaciones acerca de una forma de cuantificar la carga mental utilizando su método. No obstante, en otras ocasiones, cuando el problema a analizar no posee demasiada complejidad, estas estimaciones han demostrado ser poco útiles, tal y como se demostró en análisis preliminares hechos en este trabajo. Por ello, este análisis no será mencionado posteriormente.

Problemas sugeridos por los análisis NGOMS.

Las sugerencias de Kieras (1991) que pueden aplicarse según nuestro juicio al problema considerado son las siguientes:

• Relativos a la consistencia en el diseño.

1: los sujetos tendrán problemas a la hora de ejecutar la acción de cambiar comentario debido a que resulta inconsistente con el resto de métodos.

2: los sujetos buscaran abrir comentario en el menú archivo[23].

3: el botón de cerrar ventana y el botón de primera linea serán un problema respecto a los otros botones[24].

23Debido a problemas en el software esta predicción no pudo ser comprobada.

24 Hay ciertos botones que presentan una apariencia completamente distinta de otros, Los usuarios quizás podrían beneficiarse de una unidad en este aspecto. El ejemplo más excesivo es el botón de cerrar la

• Relativos a la eficiencia del diseño[25].

1: Abrir es un método llevado a cabo en muchas ocasiones. Parece conveniente ofrecer un método más rápido para su ejecución.

2: Seleccionar comandos de menús es más complicado que otros métodos de llevar a cabo otro tipo de acciones, tales como botones o listas. Esta dificultad se origina en el paso localizar en menú, el cual involucra una cierta complejidad. En la actualidad, muchos programas están siguiendo esta tendencia e introducen barras de iconos que permiten llevar a cabo comandos sin bajar los menús.

Es de señalar que las predicciones cualitativas de NGOMS no son numerosas en absoluto.

4.4.2 Descripción TAG

La descripción TAG realizada aquí seguirá el estilo de notación y las características establecidas en (Schiele y Green, 1990). Un ejemplo de esta descripción con comentarios puede encontrarse en el capítulo II de este trabajo. En breve, los elementos de TAG son los siguientes:

Como leer una descripción TAG (Schiele, 1990 p. 47).

• La lista de atributos y sus valores posibles listan los valores y características que son usados en TAG para describir tareas simples y listas de atributos en las reglas re-escritas.

• El diccionario de tareas simples describe las tareas simples, para las que TAG es especificado, en términos de componentes semánticos.

• Las Reglas-Tarea describen la expansión de tareas simples en primitivas y/o subtareas, la expansión de las cuales está a su vez descrita en las reglas sub-tarea.

---

ventana, ya que ha forzado finalmente a plantear una regla propia para ser afrontado. Otro botón poco claro es el de primera línea, ya que no ofrece el mismo aspecto visual que los otros botónes.

25 Estas sugerencias no son comprobables con nuestro ejercicio.

• *Las primitivas (los símbolos terminales de TAG) representan acciones del usuario.*

Atributos/valores

Efecto=MostrarTexto, CambiarFormaTexto, Salir, Ninguno, Imprimir, EscribirNuevoTexto.

TipoCambioFormaTexto=Nulo, Forma, Tamaño.

TipoTexto=Normal, Nulo, PrimeraLínea, NombreArchivo, TextoComentario

Entidad= Archivo, TextoMostrado, Programa, PrimeraLínea

TipodeBotón=Confirmar, Nulo, PrimeraLínea, CuadroCierre, Normal

Diccionario de tareas simples en TAG

1. Abrir archivo
   1. Efecto=mostrartexto
   2. Tipocambioformatexto=nulo
   3. Tipotexto=normal
   4. Entidad=Archivo
   5. Tipo de botón=Confirmar
2. Cambiar TipodeLetra
   1. Efecto=cambiarformatexto
   2. TipoCambioformatexto=Forma
   3. Tipotexto=textonormal
   4. Entidad=textomostrado
   5. Tipo de botón=nulo
3. Cambiar TamañodeLetra
   1. Efecto=cambiarformatexto
   2. TipoCambioformatexto=Tamaño
   3. Tipotexto=textonormal
   4. Entidad=textomostrado
   5. Tipo de botón=nulo
4. Salir
   1. Efecto=Salir
   2. TipoCambioformatexto=nulo
   3. Tipotexto=nulo
   4. Entidad=programa
   5. Tipo de botón=nulo
5. Ajustar Página
   1. Efecto=ninguno (visible, el supuesto efecto es ajustar página)
   2. TipoCambioformatexto=nulo
   3. Tipotexto=normal
   4. Entidad=textomostrado
   5. Tipo de botón=Confirmar
6. Imprimir
   1. Efecto=Imprimir
   2. TipoCambioformatexto=nulo
   3. Tipotexto=normal
   4. Entidad=textomostrado
   5. Tipo de botón=Confirmar

7. Cambiar PrimeraLinea
    1. Efecto=Escribirnuevotexto
    2. TipoCambioformatexto=nulo
    3. Tipotexto=Primeralinea
    4. Entidad=Primeralinea
    5. Tipo de botón=PrimeraLínea
8. Guardar Archivo con otro nombre
    1. Efecto=Escribirnuevotexto
    2. TipoCambioformatexto=nulo
    3. Tipotexto=Nombrearchivo
    4. Entidad=Archivo
    5. Tipo de botón=cuadrocierre
9. Cambiar el comentario
    1. Efecto=Escribirnuevotexto
    2. TipoCambioformatexto=nulo
    3. Tipotexto=Textocomentario
    4. Entidad=Comentarios
    5. Tipo de botón= normal

Restricciones de coocurrencia

- Si Efecto No es cambiarformatexto entonces TipoCambioformatexto es nulo
- Si Efecto No es Escribirnuevotexto entonces tipotexto es nulo

+ Primitivas

- Menú(Título de Menú-Comando de Menú):=utilizar un comando de menú
- Ratón.señalar (%posición):= poner la flecha del ratón en un punto
- Ratón.clickar (%posición):=apretar el botón del ratón y soltarlo inmediatamente.
- Ratón.liberar (%posición) := soltar el botón del ratón
- Ratón.presionar (%posición) :=apretar el botón del ratón y mantenerlo apretado
- Teclear texto:= (%caracteres)

Reglas-tarea.

+ T1 (2,3,4): Cambiartipoletra o tamaño o Salir

- T[TipoCambioformatexto=Forma/Tamaño, Efecto=Salir/ CambiarFormaTexto] := ComandoMenú [TipoCambioformatexto=Forma/Tamaño, Efecto=CambiarFormaTexto/Salir]

+ T2 (1, 5, 6): Ajustarpágina - imprimir- abrir

- T[Efecto= ajustarpágina/ imprimir/mostrartexto[26] Entidad=textomostrado/Archivo]:= ComandoMenú [Efecto=ajustarpágina/imprimir/mostrartexto] + seleccionarlista[entidad=archivo, texto mostrado]+clicarbotónresaltado

+ T3 (7): Cambiar PrimeraLinea

- T[Efecto= EscribirNuevoTexto, Tipo de texto=PrimeraLínea]:= PresionarBotón [Tipodebotón=PrimeraLínea]:+TeclearTexto (%caracteres)+ PresionarBotón [Tipodebotón=CuadroCierre]

- T4 (9): Cambiar el comentario

- T[Efecto=EscribirNuevoTexto, Tipo de texto=Textocomentario]:= ComandoMenú [Efecto=CambiarComentario]+Seleccionarlista [entidad=archivo]+PresionarBotón [Tipodebotón=Normal]+TeclearTexto[%caracteres]+PresionarBotón [Tipodebotón=Confirmar]+PresionarBotón [Tipodebotón=Normal]

- T5 (8): Guardar Archivo con otro Nombre

T[Efecto=EscribirNuevoTexto, Tipo de Texto=NombreArchivo] := ComandoMenú [Efecto=EscribirNuevoTexto, Tipo de Texto=NombreArchivo]+TeclearTexto[%caracteres]+PresionarBotón [Tipodebotón=Confirmar]

Reglas sub-tarea

- ComandoMenú [Efecto-=-cambiar-forma-texto, TipoCambioformatexto =Forma]:= Menú [Tipo-FormaCambioLetra(por ejemplo Nueva York etc.]

- ComandoMenú [Efecto=cambiarformatexto, TipoCambioformatexto =Tamaño]:= Menú [Tipo-TamañoCambioLetra(por ejemplo 14, 18 etc.]

ComandoMenú [Efecto=Salir, TipoCambioformatexto=nulo]:= Menú [Archivo-Salir]

---

26 Es interesante señalar que abrir, a pesar de parecer más cercano en la dinámica de funcionamiento a Cambiar Comentario que a Imprimir o Ajustar Página, resulta más fácil de juntar en el procedimiento con estos últimos. Es por tanto un buen lugar para un cambio de diseño que ponga más en relación a ambos.

- ComandoMenú [Efecto=ajustarpágina, TipoCambioformatexto=nulo]:= Menú [Archivo-ajustarpágina]

- ComandoMenú [Efecto=imprimir, TipoCambioformatexto=nulo]:= Menú [Archivo-imprimir]

- ComandoMenú [Efecto=CambiarComentario, TipoCambioformatexto =nulo]:= Menú [Documentos-Examinar]

- ComandoMenú [Efecto=MostrarTexto, TipoCambioformatexto=nulo]:= Menú [Archivo-Abrir]

- ComandoMenú [Efecto=EscribirNuevoTexto, TipodeTexto= Nombre Archivo]:= Menú [Archivo-GuardarComo]

- Seleccionarlista [entidad=archivo]:= Ratón.clickar (%posición)

- Seleccionarlista [entidad=Comentarios]:= Ratón.clickar (%posición)

- Seleccionarlista [entidad=textomostrado]:= nulo

- Seleccionar [Acción=clickar]:= Ratón.clickar (%posición)

- Seleccionar [Acción=liberar]:=Ratón.liberar (%posición)

- PresionarBotón [Tipodebotón=Confirmar]:= Ratón.clickar (%posiciónbotónConfirmar)

- PresionarBotón [Tipodebotón=Normal]:= Ratón.clickar (%posiciónbotónNormal)[27]1

- PresionarBotón [Tipodebotón=CuadroCierre]:= Ratón.clickar (%posiciónbotónCuadroCierre)

- PresionarBotón [Tipodebotón=PrimeraLínea]:= Ratón.clickar (%posiciónbotónPrimeraLínea)

27 Es interesante señalar los problemas que surgen de los botónes. Como es posible ver la regla que marca el uso de uno u otro botón es totalmente arbitraria. Además, cuando se trata de botones englobados dentro de una categoría más amplia (p.e. botones normales) resulta muy difícil marcar una regla que distinga entre unos y otros.

Análisis del modelo.

TAG no se presta a un análisis de tipo cuantitativo debido a que sólo comparando sistemas similares es posible derivar predicciones de este tipo (Schiele y Green, 1990). Aquí nos limitaremos a predicciones de tipo cualitativo.

Como es posible ver, dos grupos de tareas presentan una estructura común que las hace colapsarse en reglas que comparten elementos comunes. En primer lugar, cambiar el tipo de letra o el tamaño, que consisten en simplemente elegir un comando de menú apropiado, del mismo modo que Salir. Un segundo grupo implica a tareas que tienen una estructura basada en comando de menú y cuadro de diálogo. En el caso de Abrir el cuadro es más complicado (es necesario elegir un archivo) mientras que en el de Imprimir y Ajustar Página sólo es necesario apretar un botón de confirmación.

El resto de las tareas no han sido agrupadas con otras. Primera línea es la única cuya secuencia es iniciada mediante un botón, por lo que contrasta con todas las demás. Cambiar Comentario y Guardar Como en cambio podrían haber sido colapsadas entre sí del siguiente modo. Ambas son tareas que presentan una estructura del tipo: comando de menú, texto tecleado y fin de la tarea. No obstante, Cambiar comentario lleva a cabo una serie de acciones intermedias (seleccionar un archivo y apretar varios botones) que llevarían a reglas sub-tarea con valores nulos en el caso de Guardar Como. Otra posibilidad es considerar Cambiar Comentario como una tarea relacionada con Abrir Archivo pero que muestra el texto en un lugar diferente al del texto normal pero de nuevo la extensión todavía parece muy artificial28.

En definitiva, Cambiar Primera Línea, debido a su estructura peculiar, debería ser una de las tareas más difíciles de aprender a utilizar, o, al menos de descubrir como utilizar. Cambiar Comentario y Guardar Como serían también relativamente difíciles de usar mientras que el resto de las tareas, debido a su carácter más consistente deberían poder ser aprendidas a usar muy fácilmente tras pasar los primeros momentos de dificultad.

28 Este tipo de razonamientos son bastante comunes en TAG. Utilizando los artificios adecuados siempre sería posible colapsar todas las reglas en una sola llena de espacios vacios en cada ocasión. Sin embargo, este tipo de reglas es obviamente absurdo como descripción de la forma en que los usuarios percibirán y representarán el lenguaje de la tarea. Otra cuestión es que se obtenga un diseño lo suficientemente claro y preciso como para que una regla sea capaz de expresarlo.

Estas consideraciones pueden convertirse en valores ficticios tal y como se muestra en la tabla 4.4. En ella, las tareas que se consideran consistentes con una tarea anteriormente aprendida (valor cero) y deberían aprenderse más rápidamente que las consideradas como inconsistentes (valor uno). Un caso especial, Abrir Archivo, podría haber sido considerado que, tras llevarlo a cabo en la primera ocasión, es consistente consigo mismo y, por tanto, debería tener un valor de cero. No obstante, este caso no ha sido el contemplado en esta ocasión, por lo que esta tarea ha sido considerada siempre como inconsistente.

Nivel	Método de referencia	Consistencia (1=no;0=sí)
	ORDEN 1	
	Cambio de Tipo-Tamaño	
1.	Abrir archivo	1
2.	Cambiar TipodeLetra	1
3.	Cambiar TamañodeLetra	0
4.	Salir	0
	Primera Línea	
5.	Abrir archivo	1
6.	Cambiar PrimeraLinea	1
7.	Salir	0
	Cambiar Comentario	
8.	Cambiar el comentario	1
9.	Salir	0
	Duplicar	
10.	Abrir archivo	1
11.	Guardar Archivo con otro nombre	1
12.	Salir	0
	Imprimir	
13.	Abrir archivo	1
14.	Ajustar Página	0
15.	Imprimir	0
16.	Salir	0
	ORDEN 2	
	Cambio de Tipo-Tamaño	
1.	Abrir archivo	1
2.	Cambiar TipodeLetra	1
3.	Cambiar TamañodeLetra	0
4.	Salir	0
	Imprimir	
5.	Abrir archivo	1
6.	Ajustar Página	0
7.	Imprimir	0
8.	Salir	0
	Primera Línea	
9.	Abrir archivo	1
10.	Cambiar PrimeraLinea	1
11.	Salir	0

	Duplicar	
12.	Abrir archivo	1
13.	Guardar Archivo con otro nombre	1
14.	Salir	0
	Cambiar Comentario	
15.	Cambiar el comentario	1
16.	Salir	0

Tabla 4.4: Consistencias/inconsistencias extraídas a partir del análisis TAG.

4.4.3 Descripción Acción-Tarea mediante DSN (Dialogue Specification Notation)

La notación de especificación de diálogos desarrollada por Monk et al. (Curry y Monk, 1990a; Monk, 1990; Monk, 1993; Olsen, et al., 1995) permite realizar descripciones a nivel de acciones de un interfaz de usuario. Basada en reglas de producciones, su especificación ha sido semiautomatizada por medio de una hoja de cálculo por lo que no ha sido necesario en este trabajo realizar una descripción basada en texto. El lector interesado en estas descripciones puede consultarlas solicitándolas al autor.

De un modo breve, sigue una descripción del sistema utilizado.

En el lado izquierdo de la hoja de cálculo aparecen las acciones disponibles (indicadas por medio de una linea de asteriscos en la parte inferior) y las no disponibles (no asteriscos). En estas acciones se distingue entre acciones que corresponden al operador (O-) y acciones del sistema (S-).

En la parte superior de la tabla aparecen los estados del sistema o condiciones que limitan la posibilidad de que las acciones puedan ser ejecutadas o no. En estas condiciones, sólo dos estados son contemplados, la condición es verdadera o la condición es falsa.

Así pues, visto desde el punto de vista de una regla de producción, en la parte superior de la hoja de cálculo se encontraría el lado izquierdo de la regla, mientras en la parte izquierda se encontraría el lado derecho.

El vector de condiciones para cada acción está formado por todos los valores lógicos correspondiendo a cada acción. Existen dos vectores de condiciones por acción: El superior son precondiciones (que marcan si la acción puede ser disparada) y el inferior son postcondiciones (que modifican el

estado del sistema). Por ejemplo la condición botón ya señalado (segunda columna) está en estado falso (el botón no ha sido señalado todavía). Ello permite que la acción Señalar esté disponible (la linea de asteriscos aparece debajo de ella). Al ejecutarla, la postcondición asociada a ella modifica el estado del sistema, al hacer que la condición Ya-señalado pase a ser verdadera por lo que, entre otras, la acción Deseñalar pasaría a estar disponible.

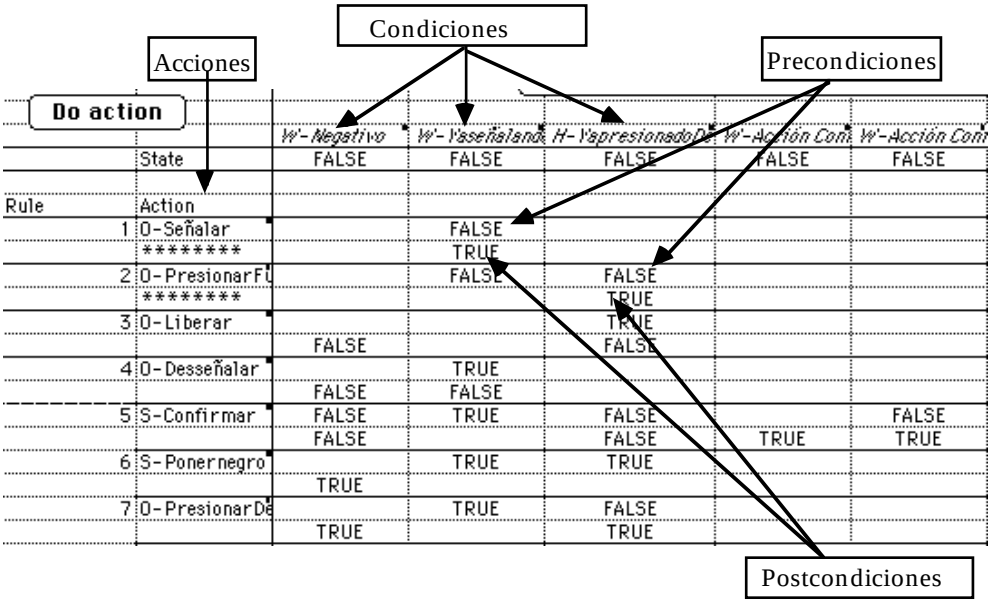


Figura 4.3: Ejemplo de descripción del comportamiento de un botón mediante una hoja de cálculo que implementa una descripción Acción-Tarea.

Elementos de la descripción realizada

La descripción realizada consta de dos niveles. En el más inferior, se describen cuatro acciones de carácter elemental (presionar botón, seleccionar de lista, teclear nuevo texto y ejecutar comando de menú) y por tanto describen elementos de interacción de nivel básico. Actualmente, estos elementos forman parte de los interfaces gráficos de muchos sistemas operativos.

En el nivel superior, las acciones corresponden específicamente a la aplicación aquí descrita. En él se incluyen menús, cuadros de diálogo, etc. que corresponden a la situación particular aquí descrita. Esta descripción utiliza los elementos del nivel anterior como subrutinas a las que hacer llamadas, de tal modo que se obvia su descripción a ese nivel.

La notación tal y como es presentada por los autores no prescribe un nivel determinado de descripción. Como muchas otras notaciones, la molaridad de las descripciones puede ser decidida en función de los intereses.[29] La decisión de considerar esos dos niveles se basa en que a la hora de realizar la especificación, parecía excesivo incluir dentro de un flujo de diálogo detalles que pertenecían a otro nivel. Por ejemplo, a la hora de seleccionar un comando de menú u otro, no parecía tener sentido describir los inconvenientes que tendrían los usuarios sobre el deseleccionar o seleccionar un comando en un menú. *Ello no significa que para los usuarios esta no sea una experiencia posible o incluso usual.* Muy probablemente, los usuarios, a la hora de realizar una especificación, experimentan dificultades a todos los niveles.

Análisis del modelo

Los modelos construidos pueden ser analizados desde dos puntos de vista. El primero de ellos es uno cuantitativo, que será descrito con más profundidad a continuación y constituye una aportación original de este trabajo. El segundo es cualitativo y se ajusta a las sugerencias llevadas a cabo por los autores que propusieron esta aproximación.

- Análisis cuantitativo.

Una aportación original de este trabajo estriba en una nueva medida de dificultad del interface extraída a partir de una descripción Acción-tarea.

Esta medida de dificultad se sustenta sobre la siguiente consideración: ante una situación dada, resulta más difícil encontrar la tarea a ejecutar cuanto mayor es el número de acciones disponibles. Así, determinando el número de acciones disponibles en cada situación, podemos estimar la dificultad que esa situación conlleva. Esta sencilla regla puede ser puesta en contacto con al menos dos puntos presentes en la literatura:

- La insistencia de ciertos conjuntos de normas (Apple, 1987) en controlar las acciones no disponibles cuando los requisitos para esa acción no se cumplen (p.e. un menú no es utilizable cuando un objeto

29 Un ejemplo proporcionado por los autores propone acciones como "alarma en central nuclear" y "corregir por medio de sistema de enfriamiento" que trascurren dentro del ámbito de un supuesto accidente. Estas acciones como vemos son tremendamente molares y por tanto aptas para un análisis de este tipo.

previo no ha sido seleccionado). Ello permite al interface sugerir "suavemente" cual es la siguiente acción a realizar.

- El modelo de Howes (1994) describe la adquisición de conocimiento acerca de un sistema de menús por medio de exploración. Este modelo parte de un examinar las posibilidades disponibles y contrastarlas con una fuente de conocimiento semántico previo relevante a la tarea. El número de posibilidades a comprobar va reduciéndose a medida que se alcanza más experiencia en el uso del sistema al no haber necesidad de comprobar todas las opciones disponibles.

En definitiva, navegar por un interface es una actividad que puede ser conceptualizada como una serie de problemas de tipo más o menos complejo que es necesario ir resolviendo hasta alcanzar el resultado deseado[30]. Una observación que soporta esta afirmación es la apreciación de que los usuarios no recuerdan las posiciones de los objetos en las pantallas, sino que necesitan de un proceso de reconocimiento hasta alcanzar el resultado deseado. Si los sujetos recordarán exactamente el elemento a utilizar, el número de acciones disponibles no influiría en el tiempo para ejecutar una tarea, pero, debido a que su tarea consiste en un explorar para seleccionar, esta característica sí que influye.

Aunque Howes utiliza un conjunto de criterios más complejo para desarrollar su modelo basado en SOAR, la aproximación presentada aquí presenta la ventaja de ser fácil de considerar a partir de una descripción Acción-Tarea. Por otro lado, resulta obvio que esta estrategia sólo permitiría captar una de las fuentes de los posibles errores, y, por tanto, otras fuentes, tal y como un inadecuado feed-back o un comportamiento inconsistente no serían detectadas en absoluto. No obstante, ello no impide la utilidad de un análisis que buscara la existencia de opciones de acción irrelevantes y que las eliminara del interfaz, así como de situaciones excesivamente "amplias" o con un grado de ambiguedad excesivo.

30 En este modelo realmente el objetivo no es un elemento tan central como en los modelos de usuario utilizados en secciones anteriores pero sin embargo siempre es más o menos necesario postular un resultado deseado (aunque sólo sea el de salir de un programa poco interesante).

Descripción operaciones.

Bajar menú: Esta acción despliega el menú correspondiente.

Comando: Abreviatura de seleccionar comando de menú. Implica mover la flecha del ratón hasta seleccionar el comando.

Seleccionar de lista: Implica click-ar sobre el elemento correcto de la lista.

Apretar botón OK: Implica clickar sobre el botón resaltado.

Presionarbotón PL: Presionar botón primera línea.

Cerrar ventana: Implica presionar sobre el botón cerrar ventana.

Presionar botón: Implica presionar sobre un botón no resaltado.

Dos cuestiones de importancia en relación con la descripción:

1) El nivel al que está realizada la descripción: El nivel al que están descritas las acciones es prácticamente el inferior posible desde el punto de vista del interfaz. No obstante, aún sería posible realizar una fragmentación mayor, en la que todas las acciones fueran partidas en un a) señalar el objeto de interés con la flecha del ratón y b) realizar la acción correspondiente sobre el botón del ratón (clickar, dobleclickar, etc.). Esto no ha sido realizado así debido a que la segunda parte tiene un tiempo excesivamente corto como para ser recogido con las herramientas a nuestra disposición, además de ser un detalle final relativamente irrelevante.

2) Examinando los valores indicados en la sección siguiente, es posible ver que para un elemento similar (por ejemplo, presionar botón OK) el valor asignado no siempre es el mismo. Ello se debe a que el contexto en que esta acción se produce es el que marca el número de opciones disponibles. Esto es importante en relación con los análisis estadísticos realizados porque los efectos encontrados no serán atribuidos simplemente a diferencias en la complejidad de especificación del comando[31].

31 Una idea interesante para controlar el efecto de esta complejidad es realizar un análisis GOMS de la complejidad de cada comando e introducir los resultados dentro del análisis de regresión para evaluar la posible confusión producida.

¿Cómo ha sido realizado este análisis?

Este análisis resulta difícil de describir sin una herramienta informática que ofrezca el comportamiento dinámico del flujo de acciones disponibles como respuesta a las acciones realizadas por el sujeto. Esta herramienta es el simulador de acciones descrito brevemente anteriormente. Mediante ella, es posible construir un modelo que ante una acción realizada muestre las acciones disponibles a continuación. Contando estas acciones disponibles es posible obtener los índices mostrados a continuación.

Acción (agrupadas por método)	**Número opciones disponibles.**
Cambiar Tipo-Tamaño.	
Bajar menú Archivo	7
Comando Abrir	10
Seleccionar de lista	11
Apretar botón OK	4
Bajar menú Tipo	9
Comando Nueva York	5
Bajar menú Tamaño	9
Comando 14	7
Bajar menú Archivo	9
Comando Salir	10
Imprimir.	
Bajar menú Archivo	7
Comando Abrir	10
Seleccionar de lista	11
Apretar botón OK	4
Bajar menú Archivo	9
Comando Ajustar Página	9
Presionar botón OK	4
Bajar menú Archivo	9
Comando Imprimir	9
Presionar botón OK	4
Bajar menú Archivo	9
Comando Salir	10
Duplicar Archivo.	
Bajar menú Archivo	7
Comando Abrir	10
Seleccionar de lista	11
Apretar botón OK	4
Bajar menú Archivo	9
Guardar Como	10
Teclear	9
Presionar botón OK	8
Bajar menú Archivo	9
Comando Salir	10
Cambiar Primera Línea.	
Bajar menú Archivo	7

Seleccionar de lista	11
Presionar botón OK	4
Presionar BotónPL	15
Teclear	18
Cerrar Ventana	17
Bajar menú Archivo	9
Comando Salir	10
Cambiar Comentario.	
Bajar menú Documento	7
Comando Examinar	3
Seleccionar de lista	7
Presionar botón	10
Teclear	10
Presionar botón	9
Presionar botón	10
Bajar menú Archivo	9
Comando Salir	10

Tabla 4.5: Número de acciones disponibles por cada acción.

Análisis cualitativo.

En esta sección se tratarán algunas de las sugerencias proporcionadas por los autores originales acerca de elementos que podrían ser evaluados mediante la notación de acción tarea.

Monk (Monk, 1990) sugiere los siguientes principios para ser analizados.

• Efectos difíciles de restituir a la situación original. Una acción produce un resultado que, en caso de no ser el deseado o por otra razón, resulta difícil de deshacer para volver a la situación original.

• Pantallas ambiguas: Una acción puede producir diferentes resultados, dependiendo del contexto en el que se produzca. Por lo tanto, conocer el contexto es necesario para poder determinar el resultado de la acción. Así, la pantalla u otro mecanismo debe de dar indicación implícita de esta situación. El diseñador puede explorar el interfaz para intentar hallar situaciones de este tipo[32].

32 Como ejemplo, el ordenador portatil utilizado para escribir este texto tiene una tecla de mayúsculas fija que no permanece bajada después de haber sido apretada así que escribir puede dar lugar a un texto en mayúsculas o en minúsculas sin recordar la historia previa. La solución de los diseñadores es una marca en la pantalla que aclara esto.

• Consistencia significa "...cosas diferentes para gente diferente"(Monk, 1990). Para él significa simplemente que "...acciones similares tienen efectos similares independientemente del contexto". La violación de este principio puede ser considerado como un exceso de modos, y se detecta cuando un usuario esperaría aplicar una serie de acciones en un contexto, pero estas se aplican en ciertos casos y no en otros. La notación acción-tarea no parece muy apropiada para realizar análisis de este tipo ya que como el mismo autor señala "...un análisis de la tarea centrado en el usuario es requerido" y señala el trabajo de Payne (Payne, 1984 #165][33] como el apropiado aquí.

• Simplicidad: Lo contrario a esto es la complejidad innecesaria, lo cual hace referencia al hecho de que podría diseñarse un conjunto de reglas más reducido que mantuviera la funcionalidad básica.

Veremos la aplicación de estos principios a nuestro caso.

1) Efectos difíciles de restituir.

Este principio no es muy importante en nuestro caso. El sistema que utilizamos para hacer el experimento no dejaba al usuario obtener resultados fuera del camino "correcto". Sólo cuando las equivocaciones eran consecutivas, el usuario recibía una pantalla indicando que había ocurrido una equivocación y qué era lo que había que hacer a continuación.

Teclear texto: Dado que los sujetos no tenían mucha experiencia usando el teclado, cuando se equivocaban al teclear algo tenían bastantes problemas en borrar lo mal escrito y volverlo a escribir bien.

2) Pantallas ambiguas.

• Apretar el botón OK para abrir un archivo sin saber si el archivo seleccionado es el correcto[34].

33 Una de las exposiciones iniciales de TAG.

34 La prueba de que ciertos diseñadores han considerado esta situación como problemática en ocasiones se puede ver en que ciertos programas ofrecen un pequeño cuadro resumen del documento cuando es seleccionado su nombre en el cuadro de diálogo de abrir.

• Seleccionar el comando Imprimir sin haberle dado al comando Ajustar Página.

• Seleccionar el botón OK sin saber si el texto escrito es el correcto (esto ocurre siempre que se escribe algo)[35].

4) Complejidad innecesaria.

La complejidad innecesaria es un criterio muy global y que resulta difícil de aplicar en un sistema experimental. Es cierto que podría pensarse en un sistema con menos elementos teniendo en cuenta la funcionalidad básica. Pero la cuestión es que esa funcionalidad es más simulada que real, por lo que resulta un poco fuera de lugar este tipo de análisis.

[35] Es necesario hacer notar que la construcción de los modelos mediante el sistema utilizado no fue de gran ayuda en este tipo de análisis. Los propios autores, (Curry, et al., en prensa), poseen una herramienta que automatiza este tipo de análisis. Sin embargo, no ha sido utilizado en esta ocasión.

Capítulo 5

Metodología

5.1. Introducción

En este capítulo se hará fundamentalmente referencia a cuestiones estadísticas, comenzando por una breve introducción al modelo estádístico utilizado, siguiendo posteriormente con un esquema comentado de los análisis llevados a cabo en el siguiente capítulo con objeto de facilitar su lectura.

Un apartado importante que podría haber sido incluido en este capítulo ha sido adelantado al comienzo del capítulo 4. Nos referimos a la descripción del programa que los sujetos aprendieron a manejar para llevar a cabo el ejercicio del que se tomaron los datos a analizar. Este apartado ha sido trasladado a aquel lugar para facilitar la lectura de las descripciones formales realizadas ya que, sin él, hubiera resultado difícil en nuestra opinión, seguir el resto del capítulo 4.

5.2. Descripción de las técnicas estadísticas utilizadas

El modelo estadístico utilizado en este trabajo ha sido el de un análisis de series temporales conjunto (Maddala, 1985; Maddala, 1987b; Moore, et al., 1994; Sayrs, 1989). Esta técnica puede ser comparada a un análisis de regresión intrasujeto que permite especificar autocorrelación (también intrasujeto), además de aprovechar en parte la variabilidad entre sujetos para aumentar la consistencia de la estimación de los parámetros calculados.

Este tipo de análisis es una técnica infrautilizada en Psicología según Moore et. al (1994) ya que, de acuerdo con una revisión realizada en bases de

datos bibliográficas, sólo encontraron nueve artículos que la utilizaran. Esta técnica es descrita habitualmente en textos de Econometría, y se aplica típicamente a situaciones en las que datos acerca de varias empresas, países o unidades son recogidos durante una serie de periodos de tiempo. De este modo, permite combinar las ventajas de datos provenientes de secciones transversales y de secciones longitudinales.

Las ventajas de este modelo en Psicología, según Moore et. al (1994), son :

1) Al utilizarse una muestra de sujetos, necesita menos observaciones por sujeto de las que se necesitaría en un análisis de series temporales ordinarias.

2) Puede proporcionar una potencia estadística considerable al analizar muestras estadísticas pequeñas, ya que, al utilizar muchas observaciones para cada sujeto, hay un considerable incremento en el N manejado.

3) El investigador aprovecha la ventaja de utilizar un análisis de regresión intrasujetos, el cual proporciona directamente tamaños de efecto útiles en la práctica aplicada en forma de coeficientes de regresión, además de indicadores de significación estadística.

Para tratar las unidades consideradas, han sido propuestos tres tipos de modelos (Maddala, 1971; Novales Cinca, 1988; Sayrs, 1989): de coeficientes constantes, de efectos fijos y de efectos aleatorios. De estos modelos, el primero de ellos introduce el efecto de las variables independientes consideradas sin tener en cuenta las distintas unidades que han producido los resultados y es equivalente a realizar un análisis de regresión basado en mínimos cuadrados entre las variables independientes y las variables dependientes. En nuestro caso, este modelo no parecía apropiado, debido a que los sujetos tuvieron comportamientos distintos, realizando las tareas con un tiempo medio diferente entre ellos. Una alternativa es el denominado en la literatura econométrica modelo de efectos fijos y que puede equipararse a un análisis de regresión intrasujeto (Pedhazur, 1982). En este modelo, cada unidad incorpora una variable ficticia (dummy). Ello equivale a otorgar una constante diferente para cada unidad en el análisis de regresión, además de considerar el efecto de las variables independientes. El modelo es también capaz de incluir constantes diferentes para cada momento serial (lo cual hubiera sido inapropiado en nuestro caso) y autocorrelación intrasujeto.

Este último modelo es considerado inapropiado por Maddala (1971, 1985, 1987) porque trata el comportamiento individual como "una ignorancia

específica" que es controlada independientemente de su origen frente a la "ignorancia general" que supone el componente de error. Esto lleva a los siguientes argumentos en contra del modelo de efectos fijos:

a) no hay razón para tratar una fuente de error de modo diferente a las otras fuentes y,

b) en la literatura del análisis de varianza se suele mencionar una perdida de generalizabilidad, ya que las inferencias que podríamos extraer sólo pueden referirse a la muestra utilizada, mientras que si quisiéramos extraer inferencias para toda la población, deberíamos utilizar un modelo aleatorio.

El modelo de efectos aleatorios en cambio no trata ese error de modo específico sino que lo añade al error general. Veamos a continuación una exposición más formal de los distintos modelos:

El modelo de coeficientes constantes puede representarse como:

$$(1)\quad y_{it} = \alpha + \beta' x_{it} + \varepsilon_{it}$$

Mientras que el modelo de efectos fijos es:

$$(2)\quad y_{it} = \alpha_i + \beta' x_{it} + \varepsilon_{it}$$

En este modelo ha sido introducida una constante para cada una de las unidades del análisis. Esto equivale a introducir una variable ficticia para cada una de los grupos (eliminando la constante general para evitar el problema de multicolinearidad perfecta que se produciría en caso de introducirla).

Y el modelo de efectos aleatorios

$$(3)\quad y_{it} = \alpha + \beta' x_{it} + \mu_i + \varepsilon_{it}$$

En este caso, los efectos individuales son considerados parte del término de error, por lo que un método de estimación según un modelo de regresión generalizada es lo apropiado.

Maddala (Maddala, 1987b) resume la solución de este modelo del siguiente modo.

Si definimos

$$(4)\quad \sum_{i,t}(y_{it} - \bar{y})^2 = \sum_{i,t}(y_{it} - \bar{y_i})^2 + \sum_{i,t}(\bar{y_i} - \bar{y})^2$$

podemos escribir estas fórmulas en los términos habituales de suma de cuadrados total, suma de cuadrados dentro (de cada sujeto a lo largo del tiempo) y suma de cuadrados entre.

$$(5)\ SCT=SCD + SCE$$

Realizando una descomposición similar para las covarianzas y el resto de las varianzas podemos llegar al siguiente estimador de la matriz de pesos:

$$(6)\ \hat{\beta} = D_{xx}^{-1} D_{xy}$$

En donde,

$$D_{xx} = (x_{it} - \bar{x}_i)^2$$
$$D_{xy} = (x_{it} - \bar{x}_i).(y_{it} - \bar{y}_i)$$

Este estimador se conoce como el estimador intra grupos y permite la solución del modelo de efectos fijos considerado anteriormente. Ahora bien, si aceptamos:

$$(7)\ \varepsilon_{it} \cong N(0, \sigma^2)$$
$$(8)\ \mu_i \cong N(0, \sigma_i^2)$$

entonces es posible derivar el siguiente estimador de mínimos cuadrados generalizados:

$$(9)\ \hat{\beta}_{GLS} = (D_{xx} + \theta E_{xx})^{-1}(D_{xy} + \theta E_{xy})$$

en donde,

$$(10)\ \theta = \frac{\sigma_\varepsilon^{\ 2}}{\sigma_\varepsilon^{\ 2} + T\sigma_\mu^{\ 2}}$$

Dos casos extremos son de importancia. Si θ=0 entonces la varianza que proviene de los grupos se anula y el modelo pasa a ser igual al modelo de efectos fijos. Si θ=1 entonces el modelo es similar a mínimos cuadrados ordinarios. Naturalmente, las varianzas del error y entre grupos deberán ser estimadas ya que en principio son desconocidas. Green (Green, 1993). expone los métodos que pueden seguirse para realizar esta estimación.

Por último, es posible demostrar que el modelo de efectos aleatorios es equivalente a utilizar mínimos cuadrados ordinarios utilizando la siguiente transformación en los datos:

$$(11)\ y_{it} - \lambda \bar{y}_i;\ \text{y},$$
$$x_{it} - \lambda \bar{x}_i.$$

en donde

$$\lambda = 1 - \sqrt{\theta}$$

5.3. Esquema de los análisis estadísticos en relación con el análisis de series temporales conjunto

En primer lugar, es necesario tener en cuenta que se ha presentado en los análisis los resultados asumiendo autocorrelación entre las distintas ejecuciones dentro de cada sujeto. El paquete estadístico utilizado realiza la transformación que corrige la autocorrelación para cada sujeto. Ello implica que ciertos datos (medias, desviaciones típicas, etc.) no tengan un valor literal ya que corresponden a las variables transformadas. Al solicitar el output del ordenador, los análisis eran realizados dos veces, la primera de ellas no consideraba la autocorrelación (aunque la calculaba en los residuales) y la segunda sí, aprovechando el valor calculado anteriormente. Los resultados de la primera regresión no serán mostrados aquí. La estimación de la autocorrelación es utilizada para realizar la transformación de Cochrane-Orcutt (Green, 1993; Ostrom, 1978) lo cual implica la pérdida de una observación por sujeto.

El esquema de los resultados es el siguiente:

1. Para cada nivel, estadísticos descriptivos de las variables dependientes consideradas, tanto en su condición original como después de realizadas las transformaciones logarítmicas que mejoraran los problemas de asimetría y no normalidad encontradas en ellas.

2. Histogramas de las variables dependientes originales y las transformadas que permiten comprobar los beneficios conseguidos por medio de las transformaciones.

3. Correlaciones entre las variables independientes. Esto permite un primer examen de la colinealidad entre ellas.

4. A continuación, para cada variable dependiente, se muestra el output del programa estadístico utilizado para realizar el análisis de series temporales conjunto. Este comprende.

a) Resultados de una regresión mínimo cuadrática de las variables independientes sobre la dependiente sin considerar los individuos. En ella se muestran:

- Estadísticos descriptivos de las variables dependientes (nº de casos, media y desviación típica), error típico y sumas de cuadrados.
- Proporción de varianza explicada no ajustada y ajustada.

• Análisis de varianza global (la significación se muestra posteriormente cuando se comparan modelos).

• Verosimilitud del modelo considerando las variables independientes y sin hacerlo (asumiendo coeficientes cero y un modelo con sólo la constante).

• Criterio de predictibilidad de Amemiya y de información de Akaike.

• Análisis de varianza global.

• Autocorrelación de los residuales con el modelo considerado.

• Indicadores para cada una de las variables independientes. Coeficientes, error típico, cocientes t, nivel de significación, media y desviación típica.

b) Resultados de una regresión mínimo cuadrática asumiendo efectos fijos para los individuos. Esto es equivalente a una regresión en la que cada individuo tiene una variable ficticia (para evitar el problema de colinealidad no se incluye la constante -Kleinbaum, et al., 1988-) y puede entenderse como un análisis de regresión intrasujeto (Pedhazur, 1982). Se muestran los mismos resultados que anteriormente añadiéndose una estimación de los efectos para cada uno de los individuos.

c) Contrastación de los distintos modelos considerados implícitamente. Estos modelos son:

1. Modelo con sólo la constante.
2. Modelo con sólo efectos de grupo.
3. Modelo con sólo variables independientes.
4. Modelo que incluye los efectos de las variables independientes y de los grupos.

Para estos cuatro modelos se muestra:

• Verosimilitud del modelo, sumas de cuadrados del error y proporción de varianza explicada.

• Pruebas de significación Chi cuadrado de verosimilitud y F de varianza entre los distintos modelos.

Este conjunto de pruebas permite valorar el interés de incluir las variables ficticias dentro del modelo frente a la regresión clásica y asimismo la significación global de los resultados.

d) Resultados del modelo de efectos aleatorios.

El output del modelo de efectos aleatorios incluye la siguiente información:

• El test de Breusch y Pagan (Green, 1993) que contrasta el modelo de regresión sin tener en cuenta los efectos de grupos (modelo 3) contra los modelos con factores. La fórmula de esta prueba es:

$$(12)\ LM = \frac{nT}{2(T-1)}\left[\frac{\sum_i (\sum_t e_{it})^2}{\sum_i \sum_t e_{it}^2} - 1\right]^2$$

$$\text{Con } H_0: \sigma_u^2 = 0 \text{ y } H_1: \sigma_u^2 \neq 0$$

Bajo la hipótesis nula este estadístico sigue la distribución Chi cuadrado con un grado de libertad. Valores bajos de este estadístico implican **no rechazar** la hipótesis nula y por tanto θ=1 en la fórmula 10 lo cual lleva a mínimos cuadrados sin efecto de grupos.

• El test de Hausman (Green, 1993) para comprobar si el modelo de efectos aleatorios es preferible al modelo de efectos fijos. Su fórmula es:

$$(13)\ H = (\hat{\beta}_{GMC} - b_{FMCO})' \left\{ Var(b_{FMCO}) - Var(\hat{\beta}_{GMC}) \right\}^{-1} (\hat{\beta}_{GMC} - b_{FMCO})$$

En esta fórmula, si la diferencia entre las varianzas de los estimadores de los pesos de la regresión obtenidos a partir del modelo de efectos fijos y del de efectos aleatorios es pequeña, H aumenta. Por el contrario cuanto menor sea la varianza del modelo basado en efectos aleatorios, mayor será el denominador y, por tanto, menor será el valor de H. La hipótesis nula corresponde a H_0:H=0 (el modelo de efectos aleatorios presenta menor variabilidad en las estimaciones de los pesos de la regresión y es el preferido) y la alternativa a H_1:H>0 (se prefiere el modelo de efectos fijos). Bajo la hipótesis nula, H sigue la distribución Chi cuadrado con grados de libertad igual al número de regresores.

• Proporción de varianza explicada por el modelo.

Al respecto, es conveniente hacer notar que el modelo de coeficientes aleatorios, a pesar de ser más eficiente en la estimación de la matriz de pesos de la regresión, en general implica un descenso en el ajuste según el criterio de la proporción de varianza explicada, debido a que la varianza de los grupos ha sido incorporada al término de error en lugar de introducirse como un factor más (Sayrs, 1989).

• Indicadores para cada una de las variables independientes. Coeficientes, error típico, cocientes t, nivel de significación, media y desviación típica, estimados según el método de coeficientes aleatorios.

e) Análisis de los residuales del análisis.

• Diagrama de dispersión de las puntuaciones predichas frente a las residuales.

• Histograma de puntuaciones residuales.

• Gráficos de series temporales por sujeto de los residuales obtenidos (un esfuerzo ha sido realizado para igualar las escalas de estos gráficos para facilitar la comparación).

• Comparación de los residuales gráficamente y por medio de análisis de varianza cuando se ha considerado apropiado en función de las cinco tareas principales que los sujetos realizaron con objeto de comprobar la variabilidad restante.

Capítulo 6

Resultados

6.1. Introducción

En este capitulo se mostrarán los resultados obtenidos acerca de la capacidad de predicción de las técnicas de descripción utilizadase en el capítulo 4 sobre el comportamiento de los sujetos en las tareas diseñadas. Este capítulo comprende los siguientes puntos: En primer lugar, comentaremos algunas cuestiones en relación con las variables dependientes utilizadas; y, en segundo, mostraremos los resultados estadísticos. Estos resultados vienen acompañados con un esquema inicial que explica los outputs del paquete estadístico utilizado y luego breves comentarios cuando estos han sido considerados convenientes. Una discusión más completa de los resultados se reserva para el capítulo posterior.

6.2. Descripción de las variables dependientes utilizadas

Revisando el capítulo II es posible ver que las características estructurales descritas en los interfaces por medio de las notaciones manejadas se relacionan con las siguientes variables dependientes del usuario en cada caso:

• NGOMS predice tiempo de aprendizaje del sistema y tiempo de ejecución de la tarea. A través del análisis de la carga mental, predice también errores, pero este aspecto de la técnica está todavía sin desarrollar profundamente.

• TAG se centra en la consistencia de las tareas, lo cual en términos de variables dependientes puede concretarse en varias posibilidades. Menor consistencia implicará mayor dificultad de aprendizaje, de recuerdo a largo plazo, de solución de problemas, mayor número de errores, etc.

• *Action-Task* indica mayor número de opciones a considerar a la hora de realizar una acción. Este factor influiría en las siguientes variables dependientes: tiempo de solución de un problema, tiempo de ejecución, errores cometidos, etc.

En nuestro caso, sólo han sido consideradas las variables dependientes descritas a continuación:

• (TSE) Tiempo necesitado para terminar la tarea descontando el tiempo empleado cuando se cometía un error (incluyendo el tiempo del propio error como la recuperación posterior en caso de haber mensajes de alarma, etc.).

• (TCE) Tiempo total necesitado para terminar la tarea (sin descontar el tiempo de los errores).

Estas variables dependientes nos parecieron suficientes ya que pueden ser vistas como un producto ponderado del resto de las otras variables más analíticas. Así, por ejemplo, mayor número de errores o mayor dificultad en ejecutar la tarea implicarán en ambos casos mayor tiempo en terminar la tarea. Además, un diseñador de interfaces no está interesado en variables analíticas, sino que le interesan más bien valores globales que reflejen el impacto conjunto de esas variables. El tiempo de realización de la tarea es este tipo de valor global, ya que es función de las otras variables dependientes.

La razón para distinguir entre la misma variable incluyendo o no los errores se encuentra en que buena parte de los modelos existentes en la literatura (fundamentalmente los derivados de GOMS) asumen que son incapaces de predecir los errores. De este modo, si hubiéramos querido seguir estrictamente los supuestos de estos modelos sólo deberíamos haber utilizado la variable dependiente tiempo sin errores. No obstante, puesto que queríamos poner a prueba otros modelos decidimos no descartar ninguna de las posibilidades.

6.3. Variables independientes utilizadas y niveles de análisis

Las variables independientes utilizadas puede clasificarse en dos tipos. Variables de control y variables derivadas de los ejercicios de modelado realizados en el capítulo anterior. Veámoslas a continuación:

• Variables de control

1) Número de serie del intento de la tarea. Como mínimo, cada tarea era intentada tres veces. En caso de que los sujetos siguieran cometiendo errores en el tercer intento, los sujetos podían intentar la tarea un número mayor de ocasiones. Esta variable predeciría la disminución del tiempo de realización de la tarea producto de la repetición de ésta.

2) Primer intento. El primer intento era siempre el más difícil de realizar y no seguía una relación lineal con el resto de los intentos. Por ejemplo, en métodos como NGOMS, que cuentan el número de reglas nuevas que el usuario tiene que aprender por método, los intentos posteriores no supondrían ninguna regla nueva. Ello hacía que en general la variable número de serie del intento (v. párrafo anterior) presentara una relación curvilinea con los datos, que significaba un desajuste para las técnicas estadísticas basadas en supuestos lineales utilizadas en este caso. Para evitar esto, se introdujo una variable ficticia (dummy) que fijaría la varianza de ese primer intento.

3) Orden de la tarea. El número de tareas realizadas anteriormente se consideró que podría afectar la velocidad con la que los sujetos resolverían la tarea siguiente independientemente de que otras variables (tales como el número de reglas nuevas) permanecieran constantes.

• Variables obtenidas a partir del análisis de los modelos.

1) NGOMS: Es el número de reglas nuevas calculado según el método NGOMS (tabla 4.3).

2) TGOMS: Es el número total de reglas calculado según el método NGOMS (tabla 4.2).

3) CC: Cambiar Comentario. Esta tarea apareció en los análisis realizados como especialmente conflictiva por lo que una variable ficticia (dummy) fue utilizada para marcarla.

4) TAG: A partir del método de Task Action Grammar se distinguió entre dos grupos de tareas, consistentes e inconsistentes. Se utilizó una variable ficticia para representar esta distinción (Tabla 4.4).

5) *Action-Task*: A partir del método de Acción Tarea se derivaron una serie de valores indicando el número de elecciones que los sujetos debían de realizar (tabla 4.5).

No obstante, los análisis han sido realizados a diversos niveles, desde un nivel más global hasta otros más detallados tal y como se encuentra descrito en el capítulo cuatro. Dependiendo del nivel considerado, los análisis incluían unas variables independientes u otras tal y como fuera apropiado. A continuación se especifican los distintos casos:

• **Nivel 1:** Este grano de análisis corresponde al nivel 2 de la descripción NGOMS. La ejecución en estas tareas pudo ser predicha mediante las variables obtenidas a partir del análisis NGOMS: TGOMS, NGOMS y CC. Además, se utilizaron las variables de control intento, primer intento y orden de la tarea.

• **Nivel 2:** Corresponde a una serie de subtareas que se incluían dentro de las cinco tareas consideradas anteriormente. En la descripción NGOMS del capítulo 3 correspondería al nivel 3. Estas subtareas son apropiadamente descritas también mediante TAG por lo que existe la oportunidad de realizar comparaciones entre las predicciones realizadas mediante los dos modelos. Las variables utilizadas fueron NGOMS, TGOMS, TAG y CC. Además se utilizaron las variables de control.

• **Nivel 3:** Corresponde a acciones básicas más elementales que las analizadas en los otros niveles. Sólo se utilizó la variables Action-Task. Además se utilizaron las variables de control.

6.4. Resultados obtenidos

1. Resultados para el Nivel 1

En el nivel 1 se manejan un total de 205 observaciones que corresponden a la combinación de 12 sujetos por 5 métodos por 3 a 6 intentos (dependiendo de la rapidez con la que consiguieran resolver el problema).

A continuación se muestran los estadísticos descriptivos y los histogramas de las variables dependientes consideradas.

V. dep.	TSE (s)	Log (TSE)	TCE	TCE*
Medias	36.11	1.511	62.12	1.677

Mediana	30.7	1.488	36.3	1.668
N	205	205	205	205
Asimetría	2.921	0.7874	3.518	0.2779
Curtosis	12.45	1.165	14.4	-0.3779
Desv. típica	20.4	0.1897	75.95	0.1008
A. Intercuartil	16.25	0.2165	29.38	0.1217
Mínimo	13.1	1.117	13.1	1.447
Máximo	174	2.241	541.3	1.914

Tabla 6.1: Estadísticos descriptivos de las v. d. en el nivel 1.

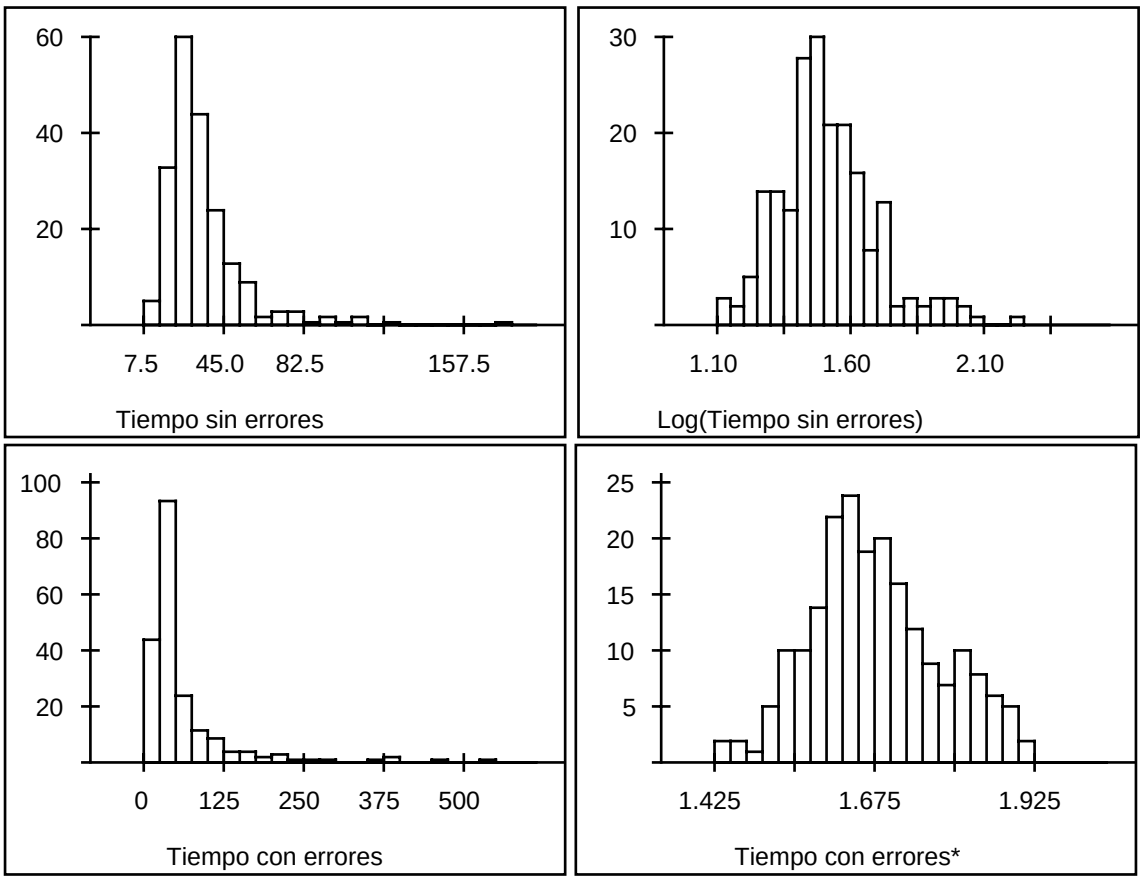


Figura 6.1: Histogramas de las variables dependientes en el nivel 1.

El tiempo medio necesario para terminar una tarea (sin descontar los errores) era un minuto aproximadamente. Si se descuenta el tiempo empleado en acciones equivocadas, este valor desciende a la mitad. Algunos sujetos llegaron a realizar alguno de los ensayos con gran velocidad (13 segundos) frente a otros ensayos en los que se necesitaron casi 10 minutos hasta que el sujeto terminó con la tarea.

Como es posible ver, existe en todas las variables dependientes una gran asimetría y curtosis, lo cual las hace poco adecuadas para la realización de los

análisis. Esta asimetría es razonable si pensamos que se trata de variables limitadas solamente por uno de los extremos y no por el otro. Además, distintos componentes vienen a conformar los resultados observados, algunos de los cuales influirán para producir valores más altos (cuando los sujetos tengan problemas para resolver la situación y se embarquen en actividades de búsqueda y exploración) que en otras ocasiones (cuando los sujetos resuelvan las tareas de una manera más directa).

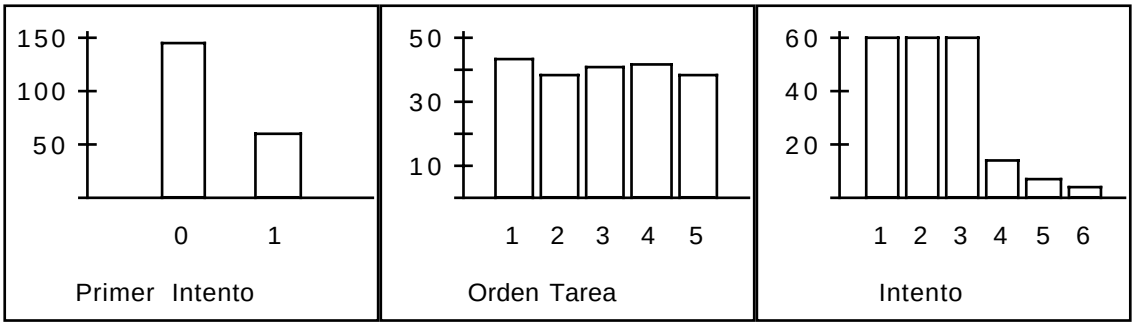
Una solución para este tipo de problemas se encuentra en realizar transformaciones de tipo logarítmico o cercanas (véase histogramas correspondientes). En nuestro caso, la transformación se realizó por medio de técnicas gráficas buscando valores que corrigieran satisfactoriamente esta situación. En el caso del tiempo sin errores la transformación logarítmica fue aceptable y en el caso de tiempo con errores la siguiente transformación (v. Hoaglin, et al., 1983) con un valor de p=-.5 fue realizada sobre la variable [1].

$$T^* = \frac{X^p - 1}{p}$$

Descripción de las variables independientes

La obtención de las reglas totales (TGOMS) y las reglas nuevas (NGOMS) pueden ser consultadas en la sección anterior. Cambiar Comentario es una variable ficticia (dummy) que hace referencia a la tarea que según los análisis realizados debería tener un comportamiento especialmente inadecuado debido a aspectos cualitativos derivados de los análisis.

A continuación se muestran los diagramas de barras de las variables independientes.



[1] Esta transformación es bastante cercana a la logarítmica. En los outputs de ordenador , esta variable aparecerá como Logcon debido a cuestiones de espacio.

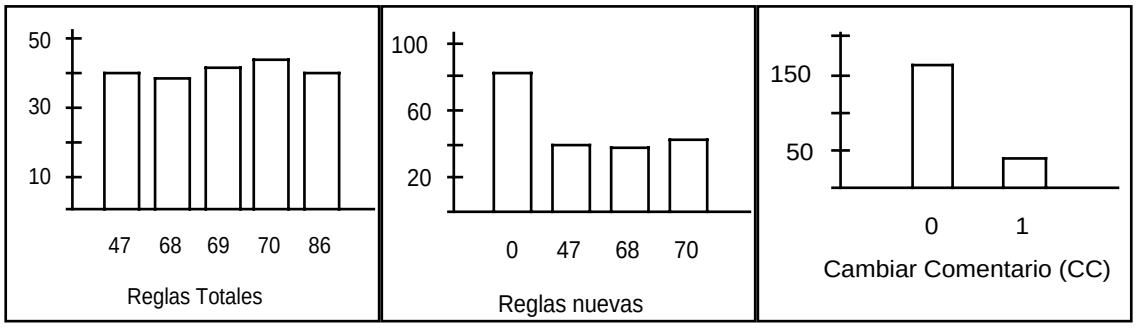


Figura 6.2: Diagramas de barras de las variables independientes al nivel 1.

Por último, se muestran las correlaciones entre las variables independientes como primer examen de la colinealidad.

Primer intento	Orden de la tarea	Intento	Reglas totales	Reglas nuevas	Cambiar Comentario
1.000					
0.015	1.000				
-0.722	-0.064	1.000			
0.001	-0.114	0.008	1.000		
-0.014	-0.782	0.052	0.070	1.000	
0.008	0.359	-0.028	-0.035	-0.209	1.000

Tabla 6.2: Correlaciones de las v. i. en el nivel 1.

La correlación más importante es entre el orden de la tarea y el número de reglas nuevas. A mayor orden de la tarea, menor era el número de reglas nuevas. También, la variable ficticia primer intento correlaciona negativamente con la variable intento.

Lo correlación entre el orden de la tarea y el número de reglas nuevas representa un inconveniente del análisis, ya que esta colinealidad oscurece los resultados alcanzados.

No obstante, es necesario tener en cuenta que esta colinealidad resulta difícil de evitar en situaciones cuasinaturales como la que estamos manejando. En la mayoría de los sistemas reales, el número de reglas nuevas disminuye a medida que nuevas funciones son aprendidas, hasta que finalmente, muy poco es aprendido en cada tarea nueva. Aunque resulta posible crear un sistema en el que cada tarea nueva aportaría un número igual de reglas nuevas (o superior), tal y como hacen Kieras y Polson (Polson, 1987), la situación que estamos manejando aquí es seguramente más cercana a las situaciones reales. Dos argumentos permiten apoyar (parcialmente) nuestro modo de hacer las cosas:

• El control de la colinealidad es inviable en las situaciones prácticas. Dado que nuestro interés no se centraba en realizar un análisis de tipo experimental en el que las variables de interés fueran aisladas de otros factores contaminantes en una situación artificial, este tipo de problemas prácticos no

fueron evitados. Nuestro interés estriba en comprobar si, a pesar de haberse producido esta situación, el modelo manejado es todavía capaz de mantener poder predictivo. Si en un experimento es posible demostrar que el número de reglas nuevas predice el tiempo, pero luego en la vida real es más sencillo utilizar el orden en que esta tarea es realizada no vemos interés en llevar a cabo costosos análisis. El argumento para defender la utilización de estos modelos pasa por demostrar que son capaces de completar las predicciones que estas variables "naturales" son capaces de hacer. Como ha sido comentado en el capítulo 1 esta postura es bastante usual entre investigadores en HCI. Un resultado que difícilmente podría producirse en la vida real es poco interesante para un área aplicada.

Podría plantearse si puede existir un interfaz en el que el orden de la tarea y el número de reglas nuevas no correlacionaran entre sí en la vida real. La respuesta es sí, pero se trataría de un interfaz que podríamos considerar malo, poco consistente, ya que no habría una reducción del número de reglas a medida que el sujeto aprende nuevas tareas.

• Puesto que nuestro interés radicaba en utilizar varios modelos de descripción para realizar un análisis de una situación semejante a una real, no se consideró conveniente ajustar la situación a alguno de los análisis que se pensaba realizar posteriormente.

Resultados para la variable dependiente tiempo sin errores

En la tabla 6.3 mostramos los resultados de un análisis de series temporales ponderadas para la variable dependiente tiempo sin errores.

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between       1.64766              11.         .149787
          Residual      4.03244             180.         .224024E-01
          Total         5.68010             191.         .297387E-01


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =            192       Weights           =  ONE
Mean of LHS        =   .1433563D+01       Std.Dev of LHS    =   .1724492D+00
StdDev of residuals=   .1373600D+00       Sum of squares    =   .3490536D+01
R-squared          =   .3854794D+00       Adjusted R-squared=   .3655490D+00
F[  6,   185]      =   .1934128D+02
Log-likelihood     =   .1122781D+03       Restr.(·=0) Log-l =   .6553442D+02
Amemiya Pr. Criter.=  -.1096646D+01       Akaike Info.Crit. =   .1955565D-01
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .2189560D+01            6.               .3649266D+00
       Residual        .3490536D+01          185.               .1886776D-01
       Total           .5680095D+01          191.               .2973872D-01
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .23648        .3172E-01    7.456   .00000   .23998        .44026
ORDTAREA  -.12424E-01    .1082E-01   -1.148   .25222   2.9558        1.3342
INTENTOV  -.32022E-02    .1200E-01    -.267   .78986   2.3004        1.1637
```

```
NGOMS      .10541E-05    .5286E-03     .002   .99841   33.623        30.583
TGOMS     -.10184E-02    .1770E-02    -.575   .56566   65.528        12.545
CC        -.86205E-01    .4988E-01   -1.728   .08552   .19499        .39342
Constant   1.5699        .1522       10.316   .00000
```

```
Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =            192       Weights           =  ONE
Mean of LHS        =   .1433563D+01       Std.Dev of LHS    =   .1724492D+00
StdDev of residuals=   .9635910D-01       Sum of squares    =   .1615603D+01
R-squared          =   .7155676D+00       Adjusted R-squared=   .6877782D+00
F[ 17,   174]      =   .2574969D+02
Log-likelihood     =   .1862314D+03       Restr.(·=0) Log-l =   .6553442D+02
Amemiya Pr. Criter.=  -.1752410D+01       Akaike Info.Crit. =   .1015555D-01
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .4064492D+01          17.               .2390878D+00
       Residual        .1615603D+01         174.               .9285075D-02
       Total           .5680095D+01         191.               .2973872D-01
Estd. Autocorrelation of e(i,t)     .025903
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .20700        .2269E-01    9.124   .00000   .23998        .44026
ORDTAREA  -.14696E-01    .7610E-02   -1.931   .05490   2.9558        1.3342
INTENTOV  -.25778E-01    .9011E-02   -2.861   .00468   2.3004        1.1637
NGOMS     -.49018E-04    .3718E-03    -.132   .89526   33.623        30.583
TGOMS     -.12063E-02    .1249E-02    -.966   .33544   65.528        12.545
CC        -.87917E-01    .3516E-01   -2.500   .01321   .19499        .39342
```

```
Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           1.62495               .11521
                    2           1.47109               .11482
                    3           1.73966               .11455
                    4           1.53151               .11482
                    5           1.63633               .11472
                    6           1.77082               .11585
                    7           1.74685               .11671
                    8           1.59784               .11487
                    9           1.77724               .11437
                   10           1.68690               .11401
                   11           1.68435               .11417
                   12           1.45941               .11479

Test Statistics for the Classical Model

       Model              Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only       65.53443          .568010D+01     .0000000
(2)  Group effects only       98.42376          .403244D+01     .2900759
(3)  X - variables only      112.27806          .349054D+01     .3854794
(4)  X and group effects     186.23136          .161560D+01     .7155676

                            Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)    65.779     11     .00000          6.686   11   179     .00000
 (3) vs (1)    93.487      6     .00000         19.341    6   185     .00000
 (4) vs (1)   241.394     17     .00000         25.750   17   174     .00000
 (4) vs (2)   175.615      6     .00000         43.382    6   174     .00000
 (4) vs (3)   147.907     11     .00000         18.357   11   174     .00000
```

```
Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .21143D-02      .23599D-01
(Used  Means.  Q  =     .0642)
Estimates:  Var[e]              =   .928508D-02
            Var[u]              =   .151847D-02
            Corr[v(i,t),v(i,s)] =   .140553
Lagrange Multiplier Test vs. Model (3) =     311.25936
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =        .00010
( 6 df, prob value = 1.000000)
Estd. Autocorrelation of e(i,t)     .015149
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .913677D-02
            Var[u]              =   .241456D-01
            Sum of Squares          .350757D+01
            R-squared               .382481D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .21584        .2256E-01    9.568   .00000   .23998        .44026
ORDTAREA  -.14484E-01    .7594E-02   -1.907   .05647   2.9558        1.3342
INTENTOV  -.19233E-01    .8844E-02   -2.175   .02965   2.3004        1.1637
NGOMS     -.84472E-04    .3721E-03    -.227   .82039   33.623        30.583
TGOMS     -.13585E-02    .1248E-02   -1.088   .27655   65.528        12.545
CC        -.91356E-01    .3512E-01   -2.601   .00930   .19499        .39342
Constant   1.6399        .1125       14.572   .00000
```

Tabla 6.3: Resultados de la variable tiempo sin errores.

El efecto de los sujetos (en el output aparecen como grupos) aparece como significativo tanto en los tests de cociente de verosimilitud como en las pruebas F. Los resultados favorecen el modelo de efectos aleatorios (Hausman=.0001;p=1) frente al de efectos fijos o al de mínimos cuadrados ordinarios. El porcentaje de varianza explicada para este modelo es R^2=.38. La autocorrelación inicialmente fue calculada en ρ=.045 para, después de controlar por ella, descender a ρ=.012.

La autocorrelación de ordenes superiores a uno, por sujeto, no alcanzaron niveles significativamente superiores a cero. Estos resultados no han sido mostrados aquí.

No obstante, en cuanto a las variables independientes consideradas, los resultados no justifican el modelo propuesto. Como puede verse en la tabla de regresión final las variables independientes número de reglas nuevas y número de reglas totales presentan coeficientes negativos, aunque no significativamente superiores a cero (NGOMS t=-.227, p=.82; TGOMS (t=-1.088, p=.276). Por otro lado, aunque la variable ficticia Cambiar Comentario (CC) aparece como significativa al 1%, su signo es el contrario al esperado.

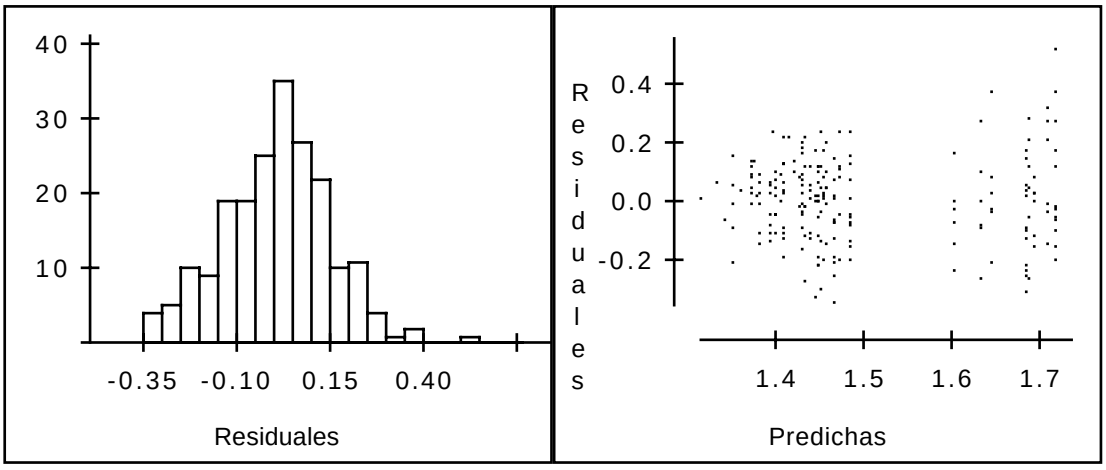
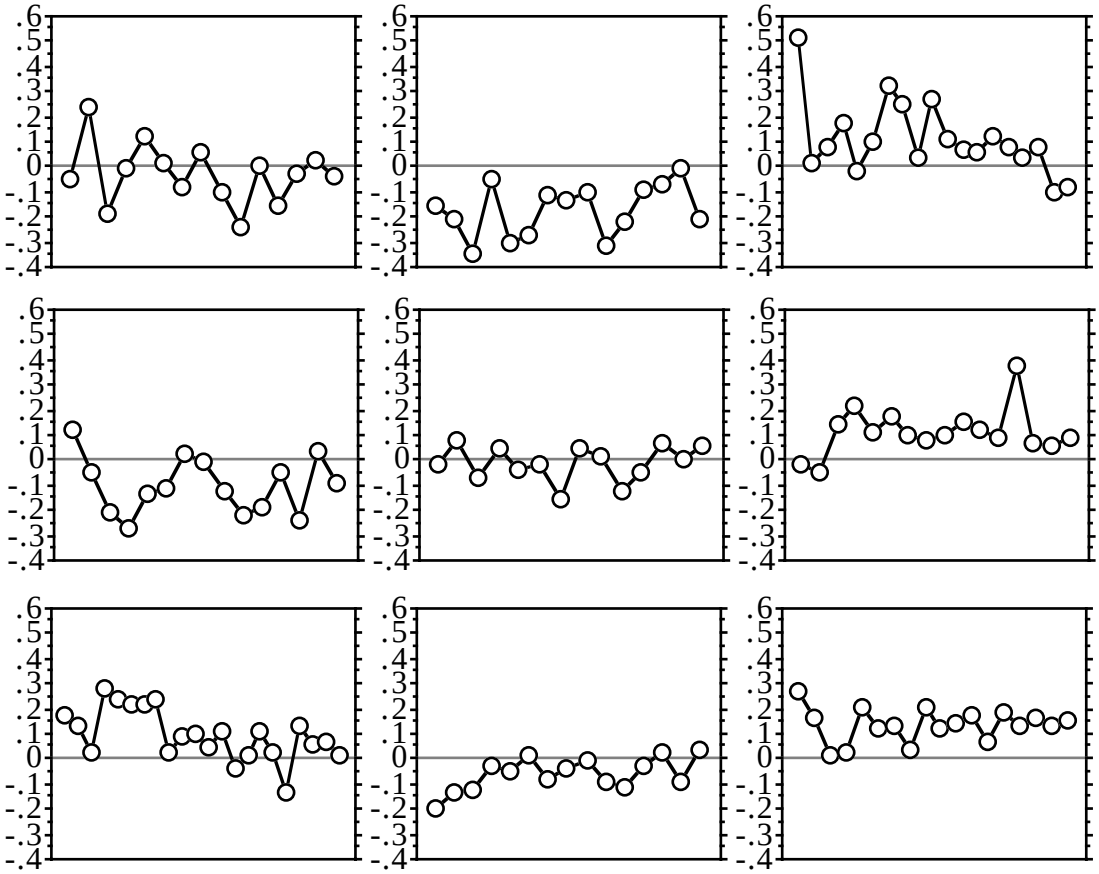


Figura 6.3: Graficos de residuales para la variable dependiente sin error.

En el diagrama de dispersión de la figura 6.3 puede apreciarse que existe una cierta heteroscedasticidad en el término de error, lo cual reafirma la idea de que nos encontramos ante un modelo mal especificado. Exploraciones de los datos no mostradas aquí revelan que muy posiblemente una transformación distinta (semejante a la realizada para la variable con error) hubiera eliminado esa heteroscedasticidad en mayor medida.



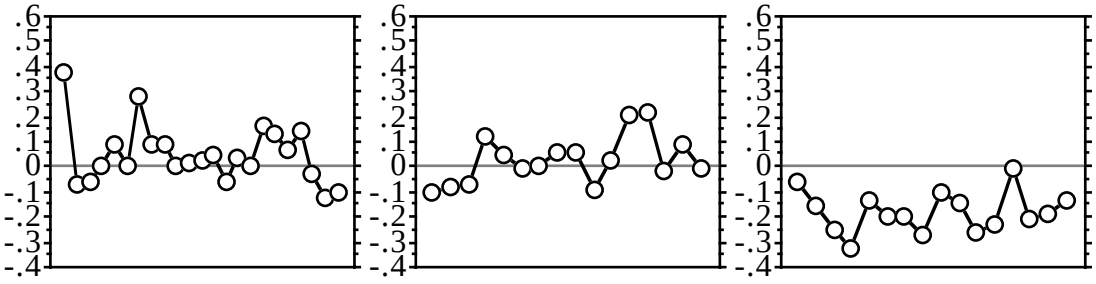


Figura 6.4: Gráficos de residuales individuales por serie

Los residuales de los sujetos muestran una aproximación visual a la calidad de los resultados. Puesto que el modelo elegido no ha sido el de efectos fijos, nada obliga a los residuales para cada individuo a distribuirse alrededor del valor de 0. Resultados no mostrados aquí revelaron como la transformación realizada eliminó la mayor parte de la autocorrelación intrasujeto para todos los individuos hasta el nivel 14[2].

Una cuestión de posible interés es si existe alguna de las tareas que fue peor explicada que el resto por el modelo propuesto. En ese caso, tendríamos una indicación de algún tipo de problema cualitativo no hallado por nuestros análisis en alguna de las tareas. El gráfico siguiente y el análisis de varianza de los residuales nos proporcionan esta información.

En los diagramas de cajas no parece que exista variabilidad especifica a alguna de las tareas que no haya sido determinada. El análisis de varianza confirma este extremo.

Fuente	gl	SC	MC	F	Prob
Const	1	0.013912	0.013912	0.65761	0.4184
Tra	4	0.046653	0.011663	0.55132	0.6983
Error	199	4.20989	0.021155		
Total	203	4.25655			

Tabla 6.4: Análisis de varianza de los residuales

Como conclusión podemos señalar que a este nivel el modelo no parece comportarse adecuadamente para predecir la variable dependiente.

[2] Hay que hacer notar no obstante que la serie no era muy larga para cada sujeto

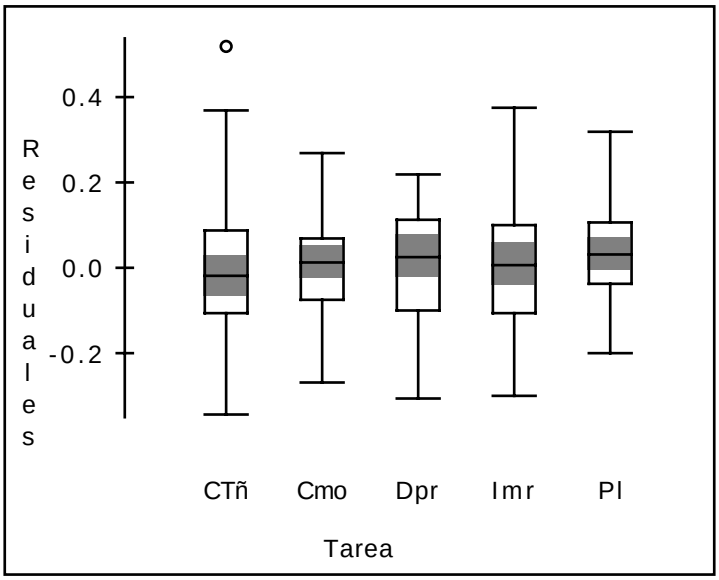


Figura 6.5: Residuales frente a tarea.

Una última comparación es la que se puede realizar entre las predicciones para el tiempo medio de aprendizaje y el tiempo medio de ejecución utilizando las sugerencias realizadas por Kieras y Elkerton mostradas en el capítulo anterior. El resultado es mostrado gráficamente y es necesario señalar que aunque el ajuste entre el aprendizaje predicho y los tiempos reales parece adecuado, la escala de las predicciones es en minutos y el tiempo es en segundos. En el caso de la comparación con la ejecución, ambos tiempos son en segundos.

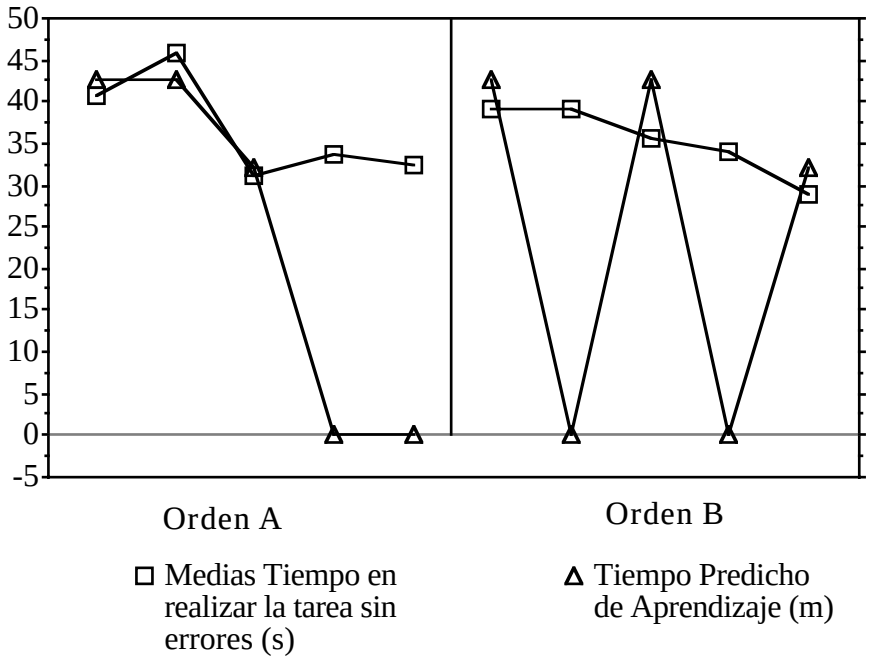


Figura 6.6: Tiempo de aprendizaje predicho frente a tiempo medio por tarea (en cada orden)

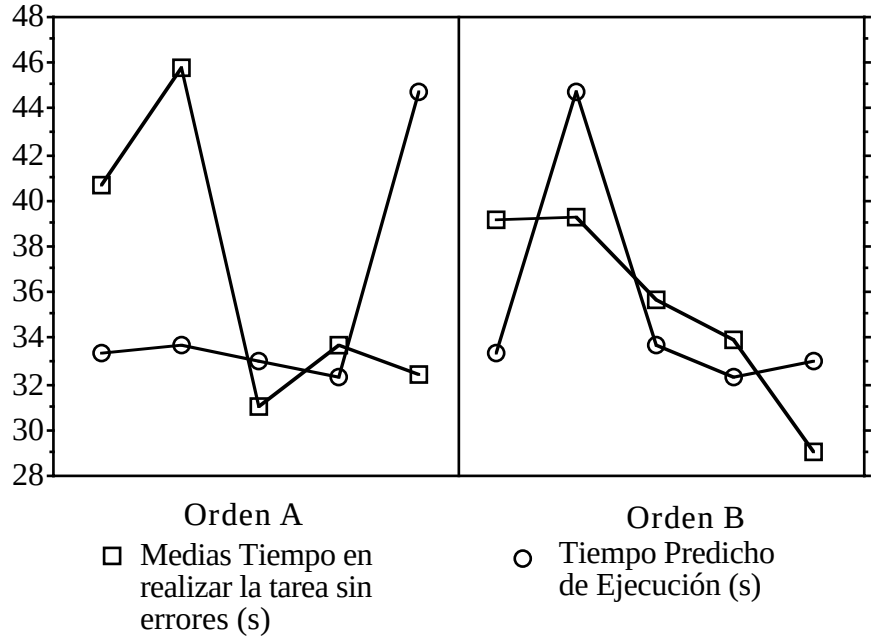


Figura 6.7: Tiempo de ejecución predicho frente a tiempo medio por tarea (en cada orden)

Resultados para la variable dependiente tiempo con errores

Pasaremos a continuación a mostrar los resultados para la variable dependiente **tiempo con errores.**

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between        .346714              11.         .315194E-01
          Residual      1.39923              180.         .777353E-02
          Total         1.74595              191.         .914109E-02


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =           192       Weights           =  ONE
Mean of LHS        =   .1454037D+01       Std.Dev of LHS    =   .9560905D-01
StdDev of residuals=   .7791866D-01       Sum of squares    =   .1123194D+01
R-squared          =   .3566855D+00       Adjusted R-squared=   .3358212D+00
F[  6,   185]      =   .1709553D+02
Log-likelihood     =   .2211304D+03       Restr.(·=0) Log-l =   .1787828D+03
Amemiya Pr. Criter.=  -.2230525D+01       Akaike Info.Crit. =   .6292668D-02
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .6227544D+00            6.               .1037924D+00
       Residual        .1123194D+01          185.               .6071318D-02
       Total           .1745948D+01          191.               .9141090D-02
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .11000        .1777E-01    6.190   .00000   .21786        .45635
ORDTAREA  -.51874E-02    .6478E-02    -.801   .42420   2.6819        1.2550
INTENTOV  -.68323E-02    .6942E-02    -.984   .32622   2.0875        1.1634
NGOMS      .30664E-03    .3163E-03     .970   .33343   30.521        28.878
TGOMS     -.11044E-02    .1065E-02   -1.037   .30093   59.462        11.745
CC        -.68746E-01    .2981E-01   -2.306   .02213   .17700        .37142
Constant   1.7911        .8221E-01   21.788   .00000
```

```
Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations      =            192       Weights           =  ONE
Mean of LHS       =   .1454037D+01       Std.Dev of LHS    =   .9560905D-01
StdDev of residuals=   .6295450D-01       Sum of squares    =   .6896088D+00
R-squared         =   .6050234D+00       Adjusted R-squared=   .5664337D+00
F[ 17,   174]     =   .1567838D+02
Log-likelihood    =   .2679599D+03       Restr.(·=0) Log-l =   .1787828D+03
Amemiya Pr. Criter.=  -.2603749D+01       Akaike Info.Crit. =   .4334825D-02
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .1056340D+01          17.               .6213762D-01
       Residual        .6896088D+00         174.               .3963269D-02
       Total           .1745948D+01         191.               .9141090D-02
Estd. Autocorrelation of e(i,t)     .037563
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .90120E-01    .1463E-01    6.160   .00000   .21786        .45635
ORDTAREA  -.53323E-02    .5246E-02   -1.016   .31068   2.6819        1.2550
INTENTOV  -.21131E-01    .5951E-02   -3.551   .00048   2.0875        1.1634
NGOMS      .31006E-03    .2561E-03    1.211   .22752   30.521        28.878
TGOMS     -.11161E-02    .8651E-03   -1.290   .19852   59.462        11.745
CC        -.69593E-01    .2418E-01   -2.878   .00444   .17700        .37142



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           1.82000               .08043
                    2           1.73552               .08044
                    3           1.88238               .08024
                    4           1.74770               .08044
                    5           1.82857               .08045
                    6           1.85840               .08124
                    7           1.89928               .08159
                    8           1.80928               .08056
                    9           1.87888               .08007
                   10           1.88136               .07972
                   11           1.83716               .07996
                   12           1.73113               .08050

Test Statistics for the Classical Model

       Model               Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only      178.78276          .174595D+01     .0000000
(2)  Group effects only      200.03451          .139923D+01     .1985818
(3)  X - variables only      221.13043          .112319D+01     .3566855
(4)  X and group effects     267.95992          .689609D+00     .6050234

                              Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)    42.503     11     .00001          4.055   11   179     .00003
 (3) vs (1)    84.695      6     .00000         17.096    6   185     .00000
 (4) vs (1)   178.354     17     .00000         15.678   17   174     .00000
 (4) vs (2)   135.851      6     .00000         29.842    6   174     .00000
 (4) vs (3)    93.659     11     .00000          9.946   11   174     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
       .10847D-02      .51907D-02
(Used  Means.  Q  =     .0642)
Estimates:  Var[e]              =   .396327D-02
```

```
            Var[u]              =   .830356D-03
            Corr[v(i,t),v(i,s)] =   .173221
Lagrange Multiplier Test vs. Model (3) =     149.26373
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =        .00010
( 6 df, prob value = 1.000000)
Estd. Autocorrelation of e(i,t)     .032675
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .368153D-02
            Var[u]              =   .536395D-02
            Sum of Squares          .109285D+01
            R-squared               .374063D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .96522E-01    .1457E-01    6.624   .00000   .21786        .45635
ORDTAREA  -.42823E-02    .5247E-02    -.816   .41447   2.6819        1.2550
INTENTOV  -.16825E-01    .5877E-02   -2.863   .00420   2.0875        1.1634
NGOMS      .31707E-03    .2560E-03    1.239   .21551   30.521        28.878
TGOMS     -.10248E-02    .8645E-03   -1.185   .23588   59.462        11.745
CC        -.67613E-01    .2417E-01   -2.798   .00515   .17700        .37142
Constant   1.8034        .7902E-01   22.822   .00000
```

Tabla 6.5: Resultados para la variable dependiente con errores.

El modelo de efectos aleatorios es preferido de nuevo (Hausman=.0001; p=1). La proporción de varianza explicada por el modelo de efectos aleatorios es R^2=.37, similar al modelo que predecía la variable dependiente sin errores. La autocorrelación inicial del modelo fue de ρ=.15 y descendió al valor de ρ=.032.

Análisis no mostrados aquí mostraron que la autocorrelación de ordenes superiores a uno por sujeto no alcanzaron niveles significativamente superiores a cero.

En este caso, el coeficiente del número de reglas nuevas sí tiene efecto positivo pero su nivel de significación no supera ninguno de los márgenes usualmente establecidos (NGOMS=.00031; t=1.23; p=.29). El número de reglas totales tampoco superó el nivel de significación del 5% (TGOMS=-.001; t=-1.185; p=.023) pero con signo negativo. La variable ficticia cambiar comentario (CC=-0.067; t=-2.79; p=.0.005) si que superó el nivel de significación pero con signo opuesto al esperado.

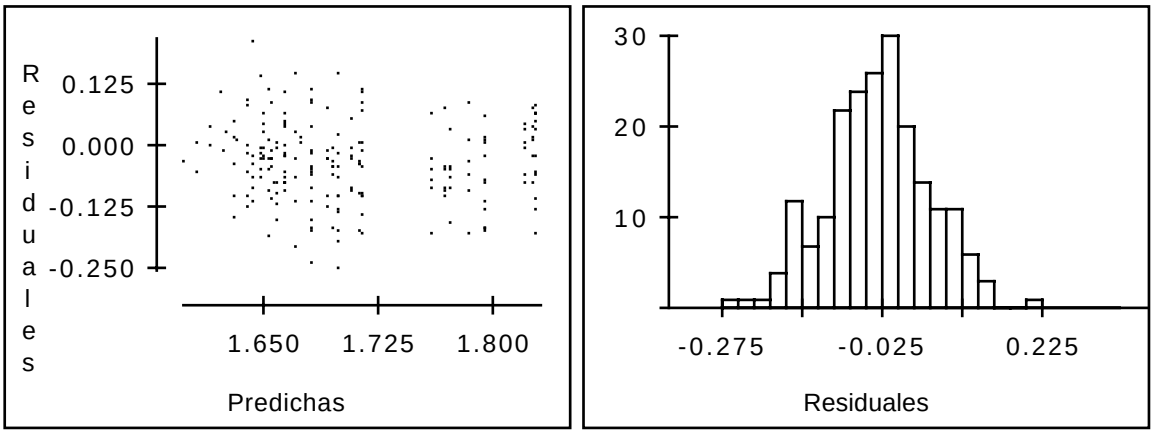


Figura 6.8: Graficos de residuales para la variable dependiente con error.

En este caso, los gráficos parecen adecuados, mostrando aproximadamente homoscedasticidad y normalidad en el error, como es deseado.

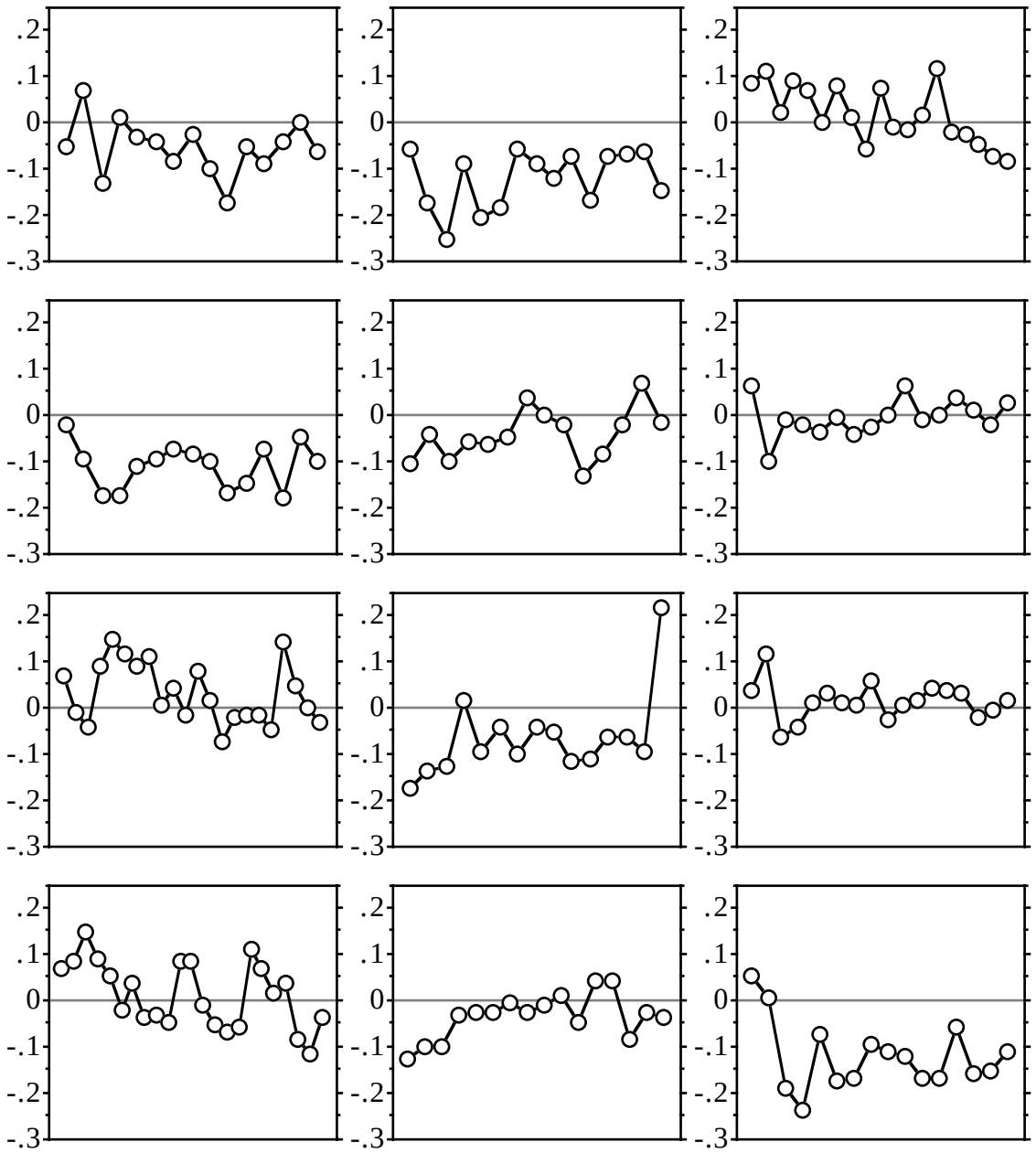


Figura 6.9: Gráficos de residuales por series temporales para la variable tiempo con error.

En los gráficos de residuales individuales es posible apreciar como algunos sujetos tuvieron residuales positivos muy altos en ocasiones (indicativo de problemas con ciertas situaciones). Otros en cambio parecen haber tenido siempre residuales negativos. La autocorrelación intrasujeto para todos los individuos fue explorada hasta el nivel 14 y no alcanzó valores significativamente superiores a cero.

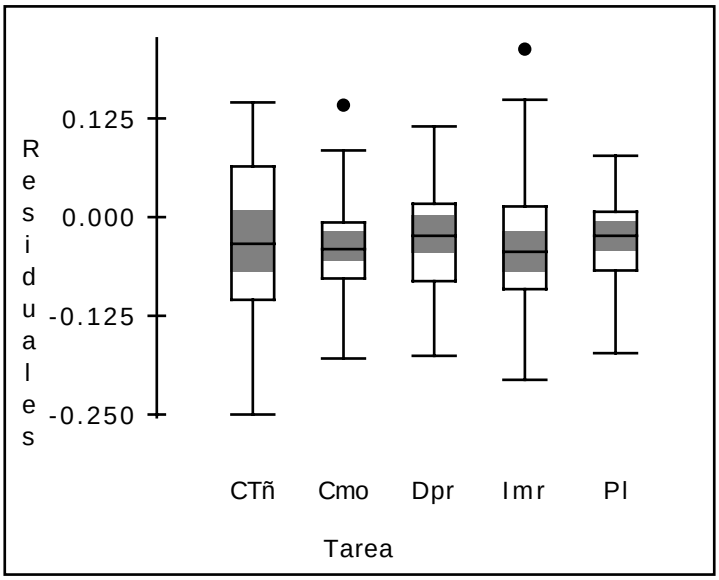


Figura 6.10: Diagrama de cajas de los residuales con respecto a las tareas.

En el gráfico que compara los residuales con las tareas no apreciamos diferencias importantes. A destacar no obstante como las medianas parecen en general tener un signo negativo, existiendo además una cierta asimetría positiva en los gráficos. Esto se ve confirmado por el análisis de varianza de la tabla 6.7[3].

Fuente	gl	SC	MC	F	Prob
Const	1	0.212504	0.212504	31.691	≤ 0.0001
Tra	4	0.001396	0.000349	0.05204	0.9949
Error	199	1.33440	0.006706		
Total	203	1.33580			

Tabla 6.7: Análisis de varianza de los residuales

De nuevo, no hay diferencia entre las tareas que aparezca como evidente.

Conclusiones respecto al nivel 1

En conclusión, los resultados en este nivel no confirman los análisis NGOMS. Ninguna de las variables realiza predicciones cuantitativas consecuentes, ni tampoco la tarea considerada como peligrosa (Cambiar Comentario) muestra diferencias con las otras. Una explicación de este resultado es que es producto de las variables introducidas para controlar el

[3] Curiosamente, los residuales no tienen una media igual a cero. En la prueba de significación de la constante se presenta un análisis de varianza para probar si la media total es diferente de cero. Este resultado es producto de los valores eliminados para calcular la autocorrelación, pero luego no utilizados para calcular los residuales.

efecto del aprendizaje. Sin estas correcciones la matriz de correlaciones entre las variables dependientes y las independientes sería la mostrada en la tabla 6.8.

LOGSIN	LOGCON	NGOMS	TGOMS
1.000			
0.855	1.000		
0.079	**0.166**	1.000	
0.092	0.048	-0.436	1.000

Tabla 6.8: Correlaciones entre NGOMS, TGOMS y las variables dependientes

Como vemos, el número de reglas nuevas presenta una relación poco importante con el tiempo de realización (valores en negrita) de las tareas. Este correlación desaparece al tener en cuenta el efecto del orden de la tarea.

2. Resultados para el Nivel 2

En el nivel 2 de análisis se manejan un total de 657 observaciones que corresponden a las 9 subtareas distintas (v. tablas 4.2 y 4.3). Los estadísticos descriptivos de las variables dependientes se muestran a continuación.

	TSE (s)	Log(TSE)	TCE (s)	Log(TCE)
Medias	12.51	2.194	20.29	2.4
Mediana	9.9	2.293	11.2	2.416
N	657	657	657	657
Asimet.	3.107	0.0067	5.839	0.544
Curt.	16.08	-0.534	46.21	0.4023
Desv.	11.64	0.8246	36.15	1.008
Interc.	11.85	1.274	14.85	1.362
Mínimo	1.2	0.1823	1.2	0.1823
Máximo	107.3	4.676	439.5	6.086

Tabla 6.9: Estadísticos descriptivos de las variables dependientes.

Se comprobó que era necesario realizar transformaciones de las variables para poder utilizarlas en los análisis realizados tal y como se hizo al nivel 1. En este caso, la transformación realizada fue hallar los logaritmos naturales de ambas variables, en lugar de los logaritmos decimales[4].

[4] Por ello ocurre la paradoja aparente de que las variables trasformadas tienen una media mayor en este nivel que al nivel uno, a pesar de que las variables originales tienen una media inferior, al ser el grano menor, en este caso.

Los histogramas de las variables antes y después de ser transformadas aparecen a continuación:

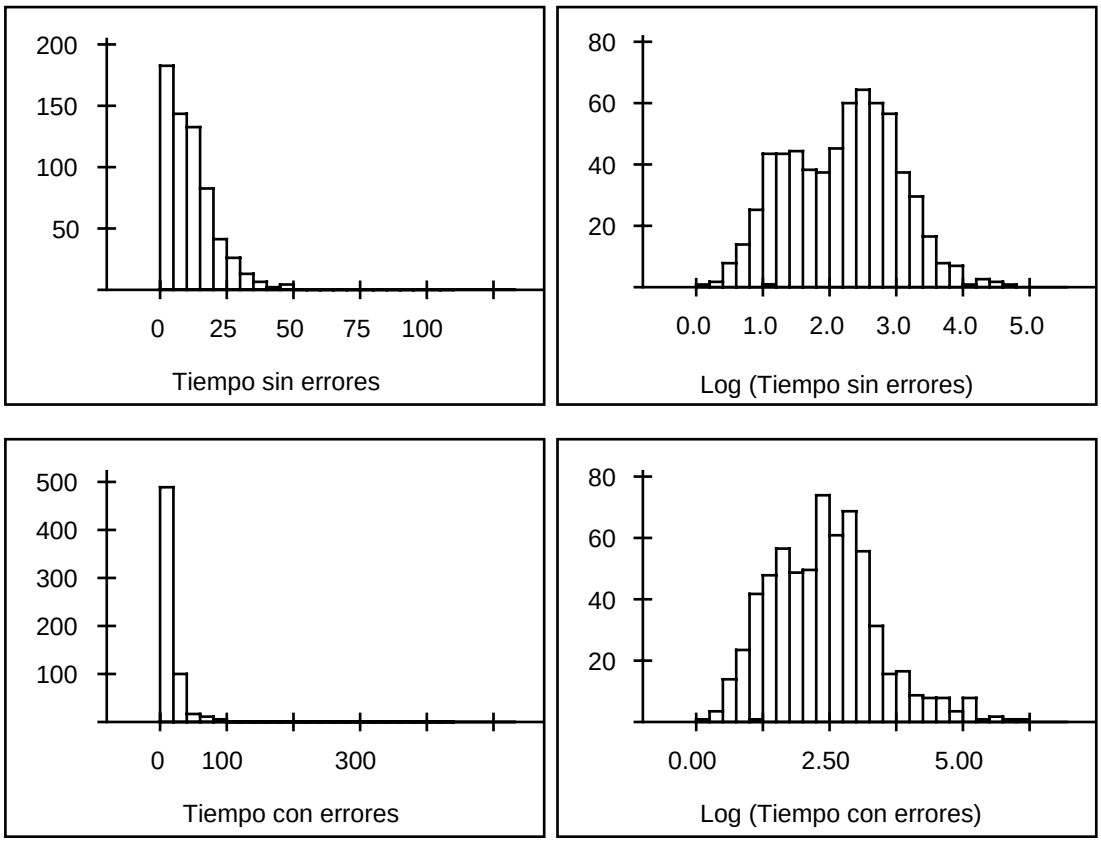


Figura 6.10: Histogramas de las variables dependientes consideradas.

Descripción de las variables independientes

A continuación se muestran los diagramas de barras de las variables independientes.

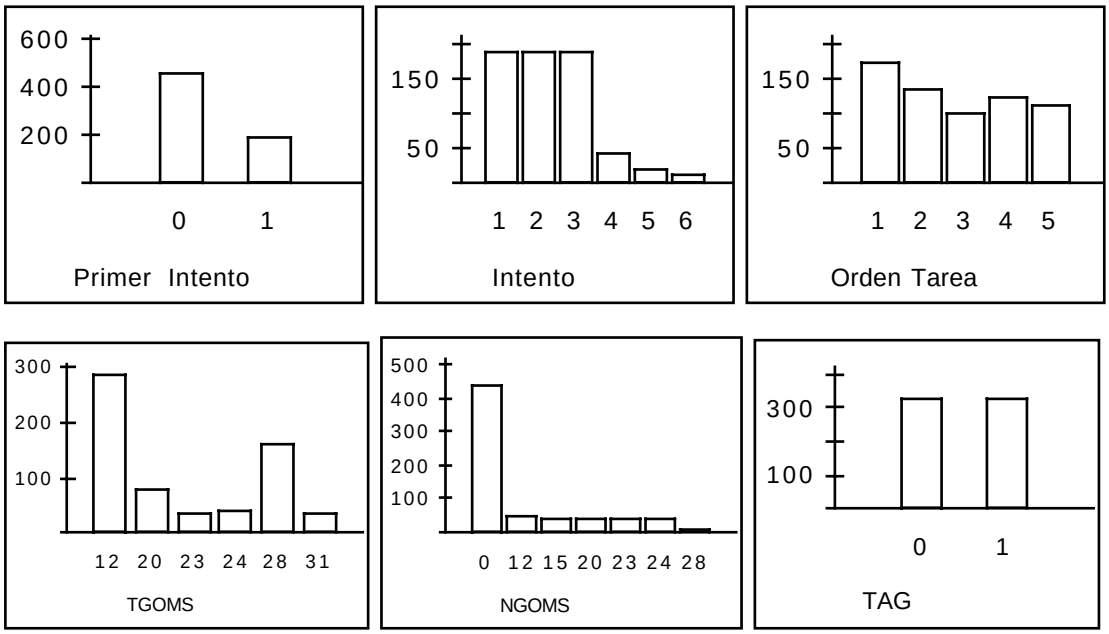


Figura 6.11: Diagramas de barras de las variables independientes a nivel 2

Como una aproximación a la colinealidad se muestra la tabla de intercorrelaciones de las variables independientes.

PRINTEN	INTENTO	ORDTAR	TAG	NGOMS	TGOMS	CAMB.CO.
1.000						
-0.716	1.000					
0.024	-0.075	1.000				
-0.002	0.004	-0.006	1.000			
0.088	-0.077	0.070	0.407	1.000		
0.008	-0.020	0.229	0.767	0.296	1.000	
0.008	-0.021	0.204	0.250	0.230	0.394	1.000

Tabla 6.10: Correlaciones entre las variables independientes utilizadas.

Como es posible ver, a este nivel, TAG está correlacionado con TGOMS y NGOMS. Esto último puede entenderse como una cierta concordancia entre los dos modos de determinar la consistencia.

Resulta interesante explicar que el orden de la tarea y el número de reglas nuevas no correlaciona en esta ocasión debido a que un mismo valor de orden puede implicar subtareas completamente nuevas y otras ya aprendidas (con número de reglas nuevas nulo). Por ejemplo, la tarea Imprimir, compuesta de cuatro subtareas (Abrir archivo, Ajustar página, Imprimir y Salir), sólo tendría reglas nuevas, en caso de estar en el orden tercero, en la segunda y tercera subtareas, mientras que la primera y la cuarta, al ser subtareas ya aprendidas anteriormente, no tendrían reglas nuevas que ser aprendidas.

Resultados para la variable dependiente tiempo sin errores

Veamos a continuación los resultados para las variable dependiente tiempo sin error.

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between       25.5595              11.         2.32359
          Residual      386.525             632.         .611590
          Total         412.084             643.         .640878


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =           644       Weights           =  ONE
Mean of LHS        =   .2237487D+01       Std.Dev of LHS    =   .8005484D+00
StdDev of residuals=   .4326360D+00       Sum of squares    =   .1190426D+03
R-squared          =   .7111208D+00       Adjusted R-squared=   .7079413D+00
```

```
F[  7,   636]      =   .2236588D+03
Log-likelihood     =  -.3701905D+03       Restr.(·=0) Log-l =  -.7700329D+03
Amemiya Pr. Criter.=   .1174505D+01       Akaike Info.Crit. =   .1894991D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .2930418D+03           7.               .4186311D+02
       Residual        .1190426D+03         636.               .1871739D+00
       Total           .4120844D+03         643.               .6408777D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .47032        .5284E-01    8.900   .00000   .28846        .45658
ORDTAREA  -.10154        .1270E-01   -7.997   .00000   2.9132        1.4926
INTENTOV   .57671E-02    .1989E-01     .290   .77189   2.4277        1.2168
NGOMS      .27662E-01    .2131E-02   12.980   .00000   6.1186        8.9897
TAG2      -.26947        .5856E-01   -4.602   .00000   .50802        .49648
TGOMS      .92718E-01    .4142E-02   22.383   .00000   20.003        7.1925
CC         .45864E-01    .7990E-01     .574   .56594   .62500E-01    .23848
Constant   .47893        .8949E-01    5.352   .00000



Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2237487D+01       Std.Dev of LHS    =   .8005484D+00
StdDev of residuals=   .3818547D+00       Sum of squares    =   .9113311D+02
R-squared          =   .7788484D+00       Adjusted R-squared=   .7724793D+00
F[ 18,   625]      =   .1222842D+03
Log-likelihood     =  -.2841649D+03       Restr.(·=0) Log-l =  -.7700329D+03
Amemiya Pr. Criter.=   .9415058D+00       Akaike Info.Crit. =   .1501149D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .3209513D+03          18.               .1783063D+02
       Residual        .9113311D+02         625.               .1458130D+00
       Total           .4120844D+03         643.               .6408777D+00
Estd. Autocorrelation of e(i,t)    -.031269
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .41397        .4756E-01    8.703   .00000   .28846        .45658
ORDTAREA  -.10554        .1129E-01   -9.352   .00000   2.9132        1.4926
INTENTOV  -.37927E-01    .1883E-01   -2.014   .04397   2.4277        1.2168
NGOMS      .27599E-01    .1882E-02   14.663   .00000   6.1186        8.9897
TAG2      -.26221        .5189E-01   -5.053   .00000   .50802        .49648
TGOMS      .92247E-01    .3670E-02   25.132   .00000   20.003        7.1925
CC         .54404E-01    .7056E-01     .771   .44069   .62500E-01    .23848



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           .52790              .09232
                    2           .23497              .09175
                    3           .75673              .09285
                    4           .38417              .09175
                    5           .62119              .09227
                    6           .87876              .09292
                    7           .78674              .09547
                    8           .55337              .09238
                    9           .85927              .09122
                   10           .67421              .09507
                   11           .63901              .09189
                   12           .24482              .09150

Test Statistics for the Classical Model

       Model              Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -770.03289          .412084D+03     .0000000
(2)  Group effects only     -749.41465          .386525D+03     .0620248
```

```
(3)  X - variables only     -370.19051          .119043D+03     .7111208
(4)  X and group effects    -284.16485          .911331D+02     .7788484

                              Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)    41.236     11     .00002          3.799   11   631     .00003
 (3) vs (1)   799.685      7     .00000        223.659    7   636     .00000
 (4) vs (1)   971.736     18     .00000        122.284   18   625     .00000
 (4) vs (2)   930.500      7     .00000        289.404    7   625     .00000
 (4) vs (3)   172.051     11     .00000         17.401   11   625     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .31685D-02      .13208D+00
(Used  Means.  Q  =     .0191)
Estimates:  Var[e]              =   .145813D+00
            Var[u]              =   .385985D-03
            Corr[v(i,t),v(i,s)] =   .002640
Lagrange Multiplier Test vs. Model (3) =     690.18156
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =        .00010
( 7 df, prob value = 1.000000)
Estd. Autocorrelation of e(i,t)    -.024585
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .146852D+00
            Var[u]              =   .130219D+00
            Sum of Squares          .119034D+03
            R-squared               .711142D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .46383        .4675E-01    9.921   .00000   .28846        .45658
ORDTAREA  -.10200        .1122E-01   -9.094   .00000   2.9132        1.4926
INTENTOV   .64555E-03    .1772E-01     .036   .97094   2.4277        1.2168
NGOMS      .27651E-01    .1881E-02   14.699   .00000   6.1186        8.9897
TAG2      -.26790        .5172E-01   -5.180   .00000   .50802        .49648
TGOMS      .92608E-01    .3659E-02   25.310   .00000   20.003        7.1925
CC         .47129E-01    .7052E-01     .668   .50396   .62500E-01    .23848
Constant   .49347        .7706E-01    6.404   .00000
```

Tabla 6.11: Resultados para la variable dependiente tiempo sin errores.

Los resultados favorecen el modelo de efectos aleatorios (Hausman=.00010;p=1) con un coeficiente de determinación de R^2=0.71. Todos los factores de interés aparecen con niveles de significación menores a .01 salvo TAG .

Las reglas nuevas tienen un coeficiente de b=.027. Para apreciar el efecto de este coeficiente podemos ver el lado izquierdo de la figura 5.12, que toma como ordenada los valores utilizados en NGOMS (entre 0 a 23), y como abscisa los valores predichos en puntuaciones directas asumiendo valores medios en el resto de variables independientes. El resultado es bastante lineal e implica un efecto medio de aproximadamente 0.77 s. por cada regla nueva.

Las reglas totales tienen un coeficiente de b=.092. El efecto en puntuaciones directas de este coeficiente puede ser apreciado en la parte derecha de la figura 6.12. Como es posible ver, el crecimiento, asumiendo valores

medios en el resto de variables independientes, es curvilíneo. Por ello, valores muy altos en reglas totales significará una duración en el tiempo para realizar las tareas de gran importancia.

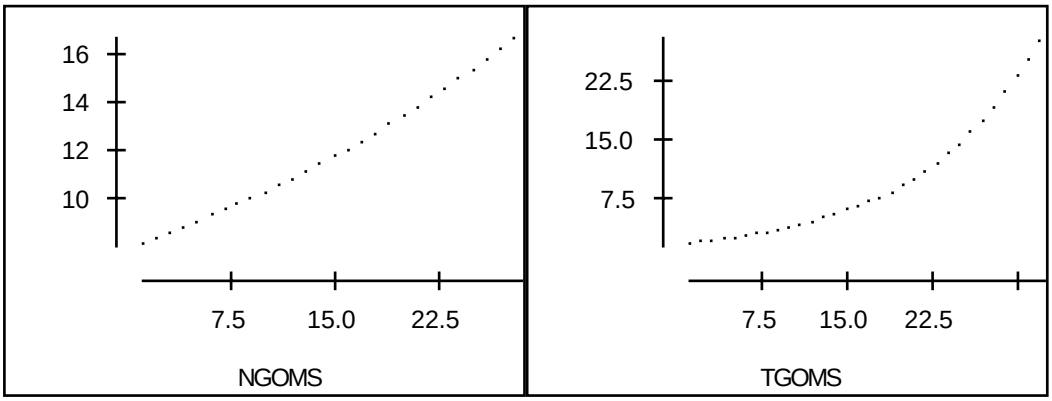


Figura 6.12: Influencia de NGOMS y TGOMS asumiendo valores medios en el resto de las variables independientes en la escala original.

TAG tiene un coeficiente significativo al nivel del 5% pero su signo es negativo. Este signo es debido a la correlación con las otras variables independientes, ya que el signo de la correlación simple entre el tiempo de realización y la variable extraída del análisis TAG es positiva. En la tabla 6.14 es posible ver un análisis que utiliza sólo la variable TAG en donde es posible comprobar este efecto.

La correlación con las otras variables ha cambiado su efecto. La variable ficticia Cambiar Comentario no alcanza un efecto significativamente superior a cero.

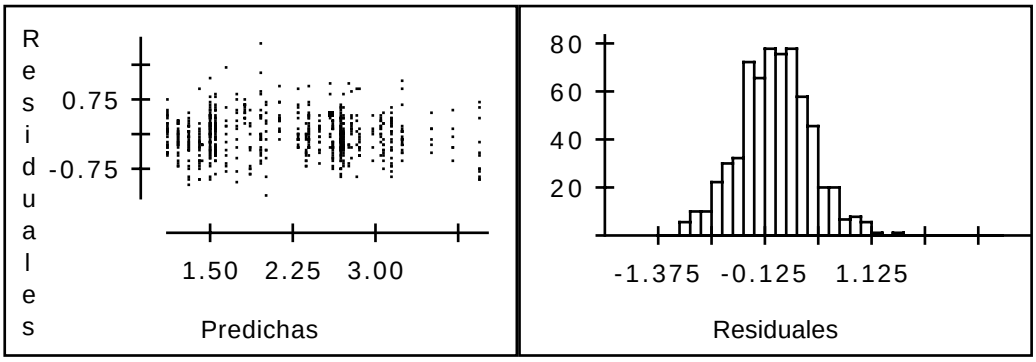


Figura 6.14: Gráficos de residuales de la variable dependiente tiempo sin error.

Los gráficos de residuales presentan un aspecto correcto.

Tanto el sujeto 3 como el sujeto 6 presentan residuales excepcionales. El primero un residual positivo y el segundo negativo. Por lo demás, todos los sujetos parecen bien ajustados por las ecuaciones calculadas. Resultados no mostrados aquí revelaron como la transformación realizada eliminó la mayor

parte de la autocorrelación intrasujeto para todos los individuos hasta el nivel 14.

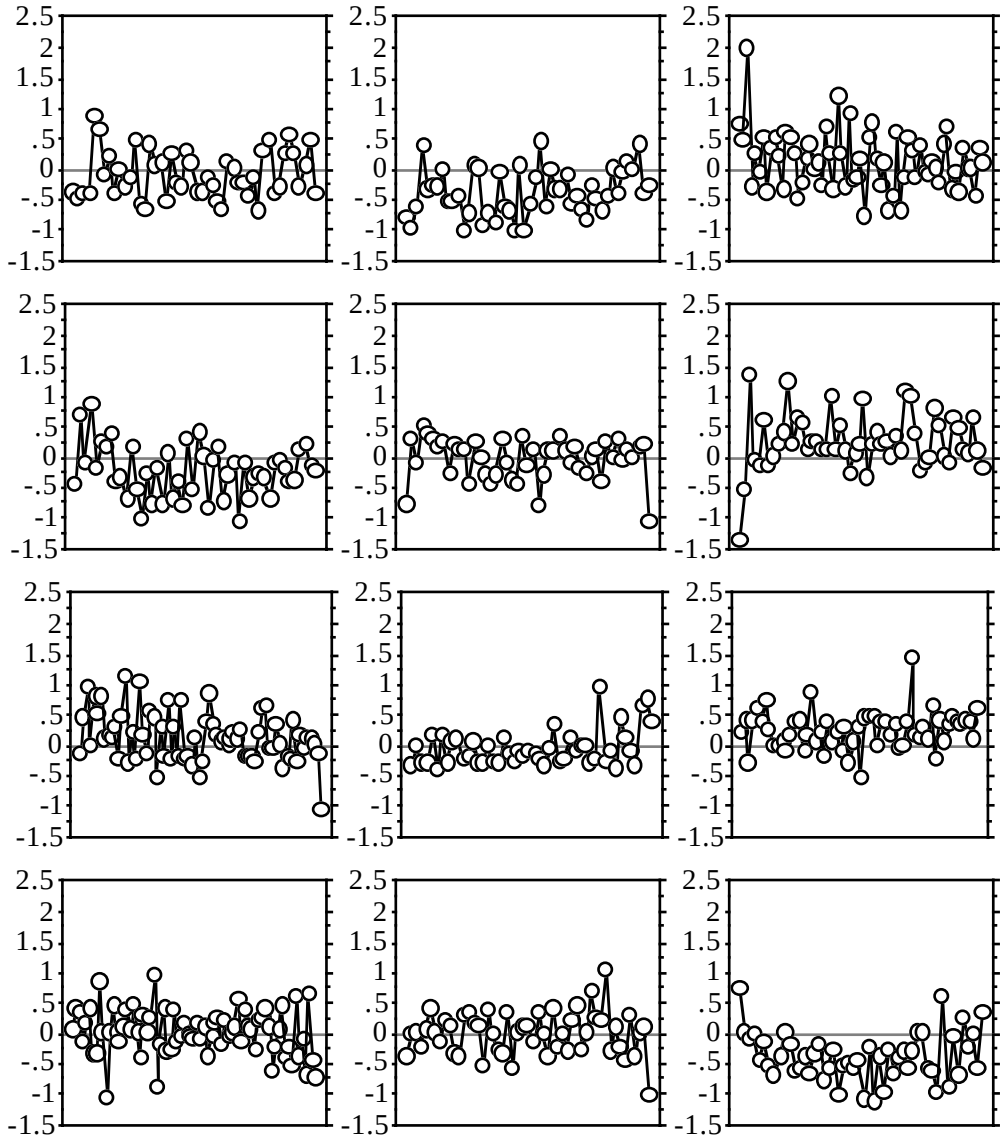


Figura 6.15: Gráficos de residuales por series temporales para la variable tiempo sin error.

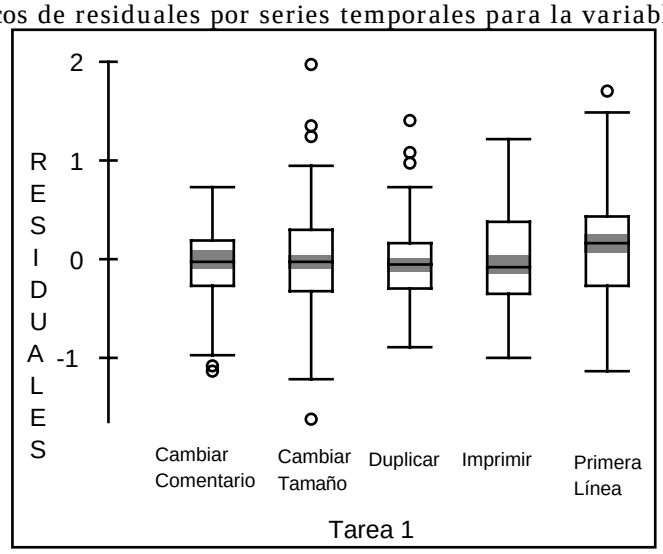


Figura 6.16: Gráfico de cajas para la tarea 1.

Los diagramas de cajas revelan sólo ligeras diferencias entre las tareas. El análisis de varianza de la tabla 6.12 muestra que no existen diferencias significativas entre las tareas.

Fuente	gl	SC	MC	F	Prob
Const	1	0.018835	0.018835	0.09947	0.7526
T1	4	0.422290	0.105572	0.55755	0.6936
Error	651	123.268	0.189352		
Total	655	123.690			

Tabla 6.12: Análisis de varianza para los residuales.

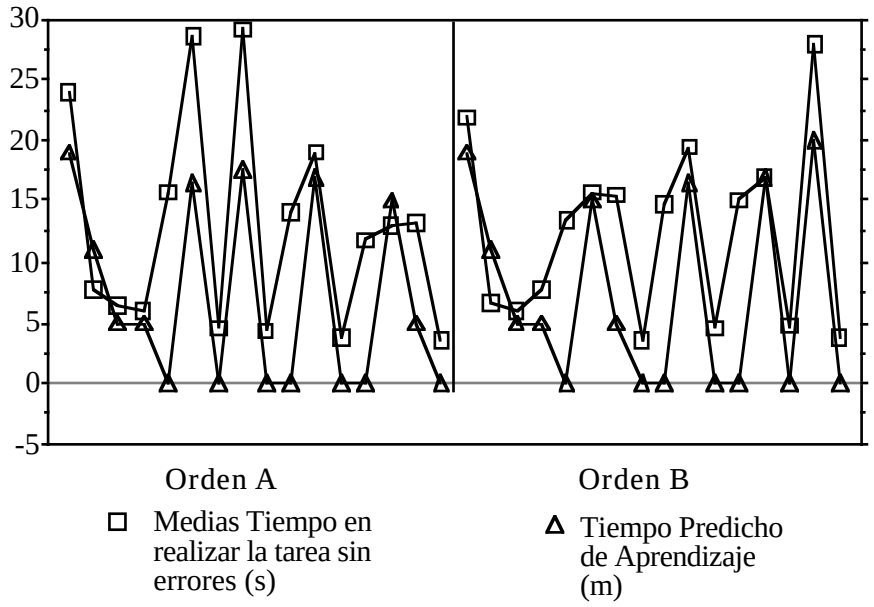


Figura 6.17: Tiempo de aprendizaje predicho frente a tiempo medio por tarea (en cada orden)

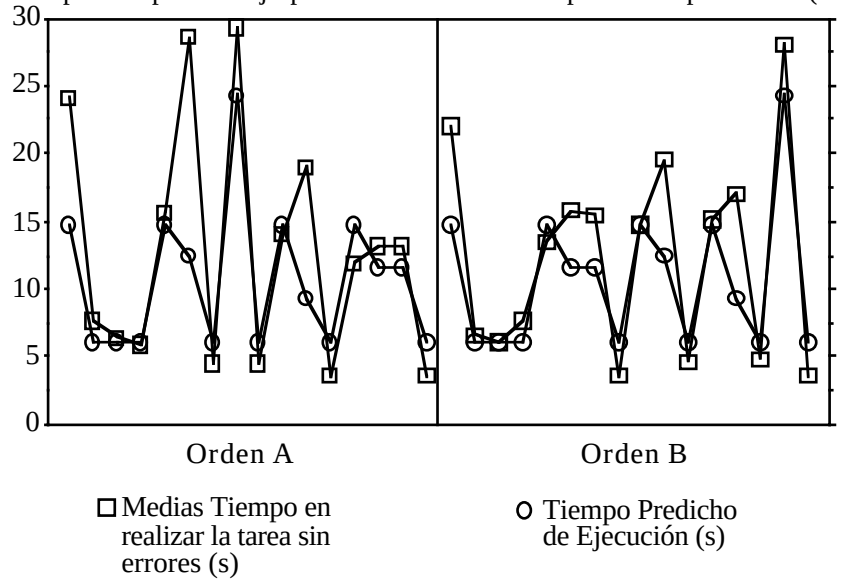


Figura 6.18: Tiempo de aprendizaje predicho frente a tiempo medio por tarea (en cada orden)

Como es posible ver, de nuevo las predicciones realizadas mediante NGOMS no ajustan en cuanto al tiempo de aprendizaje en nuestro caso (ya que la escala es de nuevo diferente, minutos predichos y segundos en cuanto a ejecución) pero sí que se aproximan mucho en cuanto a la ejecución.

Comparación entre modelos

En el análisis anterior se han utilizado dos métodos para analizar formalmente un interfaz. De estos dos métodos, es posible observar que uno de ellos (TAG) ofrece unos resultados contradictorios. Esto es debido probablemente a la colinealidad entre este método y una de las variables extraidas del otro método de análisis (en concreto con la variable TGOMS). Una situación más común a la hora de realizar un diseño podría ser aplicar un único método de análisis en lugar de ambos con objeto de reducir la cantidad de trabajo realizada. A continuación mostramos los resultados utilizando las variables de uno y otro método de análisis para luego realizar una comparación entre las proporciones de varianza explicadas[5].

Resultados para la variable dependiente tiempo sin errores utilizando NGOMS

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between       25.5595              11.         2.32359
          Residual      386.525             632.         .611590
          Total         412.084             643.         .640878


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2237487D+01       Std.Dev of LHS    =   .8005484D+00
StdDev of residuals=   .4394743D+00       Sum of squares    =   .1230287D+03
R-squared          =   .7014478D+00       Adjusted R-squared=   .6986357D+00
F[  6,   637]      =   .2494384D+03
Log-likelihood     =  -.3807959D+03       Restr.(·=0) Log-l =  -.7700329D+03
Amemiya Pr. Criter.=   .1204335D+01       Akaike Info.Crit. =   .1952370D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .2890557D+03           6.               .4817595D+02
       Residual        .1230287D+03         637.               .1931376D+00
       Total           .4120844D+03         643.               .6408777D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .46949        .5367E-01    8.748   .00000   .28846        .45658
ORDTAREA  -.82948E-01    .1223E-01   -6.785   .00000   2.9135        1.4922
INTENTOV   .47017E-02    .2021E-01     .233   .81602   2.4279        1.2166
NGOMS      .24290E-01    .2033E-02   11.948   .00000   6.1186        8.9897
TGOMS      .78214E-01    .2731E-02   28.638   .00000   20.003        7.1918
CC         .88530E-01    .8061E-01    1.098   .27212   .62500E-01    .23848
Constant   .59513        .8703E-01    6.838   .00000
```

[5] En estos resultados se ha procurado que la transformación debida al control de la autocorrelación sea la misma para los tres análisis. De este modo se hace posible realizar las comparaciones entre las proporciones de varianza explicadas por las distintas ecuaciones.

```
Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2237487D+01       Std.Dev of LHS    =   .8005484D+00
StdDev of residuals=   .3892943D+00       Sum of squares    =   .9487035D+02
R-squared          =   .7697793D+00       Adjusted R-squared=   .7635273D+00
F[ 17,   626]      =   .1231253D+03
Log-likelihood     =  -.2971060D+03       Restr.(·=0) Log-l =  -.7700329D+03
Amemiya Pr. Criter.=   .9785902D+00       Akaike Info.Crit. =   .1557859D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .3172140D+03           17.               .1865965D+02
       Residual        .9487035D+02          626.               .1515501D+00
       Total           .4120844D+03          643.               .6408777D+00
Estd. Autocorrelation of e(i,t)    -.026909
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .41328        .4848E-01    8.524   .00000   .28846        .45658
ORDTAREA  -.87300E-01    .1090E-01   -8.010   .00000   2.9135        1.4922
INTENTOV  -.38872E-01    .1919E-01   -2.025   .04285   2.4279        1.2166
NGOMS      .24314E-01    .1801E-02   13.501   .00000   6.1186        8.9897
TGOMS      .78100E-01    .2421E-02   32.261   .00000   20.003        7.1918
CC         .95856E-01    .7145E-01    1.342   .17974   .62500E-01    .23848



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1            .63817               .09153
                    2            .35138               .09058
                    3            .86236               .09228
                    4            .50057               .09058
                    5            .73883               .09107
                    6            .99215               .09197
                    7            .91486               .09389
                    8            .66395               .09154
                    9            .97113               .09027
                   10            .77934               .09462
                   11            .75348               .09084
                   12            .35212               .09078

Test Statistics for the Classical Model

       Model               Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -770.03289          .412084D+03     .0000000
(2)  Group effects only     -749.41465          .386525D+03     .0620248
(3)  X - variables only     -380.79586          .123029D+03     .7014478
(4)  X and group effects    -297.10602          .948703D+02     .7697793

                              Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)    41.236     11     .00002          3.799   11   631     .00003
 (3) vs (1)   778.474      6     .00000        249.438    6   637     .00000
 (4) vs (1)   945.854     17     .00000        123.125   17   626     .00000
 (4) vs (2)   904.617      6     .00000        320.746    6   626     .00000
 (4) vs (3)   167.380     11     .00000         16.891   11   626     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .85560D-02      .10642D+00
(Used  Means.  Q  =     .0191)
Estimates:  Var[e]             =   .151550D+00
```

```
           Var[u]              =   .566401D-02
           Corr[v(i,t),v(i,s)] =   .036027
Lagrange Multiplier Test vs. Model (3) =     659.75064
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =        .00010
( 6 df, prob value = 1.000000)
Estd. Autocorrelation of e(i,t)    -.025330
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .151633D+00
           Var[u]              =   .107763D+00
           Sum of Squares          .123470D+03
           R-squared               .700378D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .43340        .4817E-01    8.997   .00000   .28846        .45658
ORDTAREA  -.85737E-01    .1087E-01   -7.884   .00000   2.9135        1.4922
INTENTOV  -.23498E-01    .1877E-01   -1.252   .21063   2.4279        1.2166
NGOMS      .24312E-01    .1801E-02   13.500   .00000   6.1186        8.9897
TGOMS      .78118E-01    .2420E-02   32.275   .00000   20.003        7.1918
CC         .93359E-01    .7144E-01    1.307   .19126   .62500E-01    .23848
Constant   .66971        .7890E-01    8.489   .00000
```

Tabla 6.13: Resultados para la variable tiempo de realización sin errores utilizando sólo el modelo NGOMS.

Como es posible ver, la ecuación sólo incluyendo las variables NGOMS y TGOMS explica prácticamente la misma varianza que cuando se incluye también la variable TAG (R^2=.700 para la ecuación sin TAG y R^2=0.711 con ella). Podemos por tanto afirmar que introducir esta variable no aporta nada a la ecuación calculada sin ella.

Resultados para la variable dependiente tiempo sin errores utilizando TAG

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between       25.5595              11.         2.32359
          Residual      386.525             632.         .611590
          Total         412.084             643.         .640878


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2237487D+01       Std.Dev of LHS    =   .8005484D+00
StdDev of residuals=   .6010569D+00       Sum of squares    =   .2304899D+03
R-squared          =   .4406731D+00       Adjusted R-squared=   .4362896D+00
F[  5,   638]      =   .1005313D+03
Log-likelihood     =  -.5829441D+03       Restr.(·=0) Log-l =  -.7700329D+03
Amemiya Pr. Criter.=   .1829019D+01       Akaike Info.Crit. =   .3646353D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .1815945D+03           5.               .3631890D+02
       Residual        .2304899D+03         638.               .3612694D+00
       Total           .4120844D+03         643.               .6408777D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .44567        .7338E-01    6.074   .00000   .28846        .45658
ORDTAREA   .14289E-01    .1634E-01     .875   .38179   2.9132        1.4926
INTENTOV  -.31247E-03    .2764E-01    -.011   .99098   2.4277        1.2168
TAG2       .85952        .4932E-01   17.426   .00000   .50802        .49648
CC         .68407        .1049        6.523   .00000   .62500E-01    .23848
Constant   1.5408        .1047       14.717   .00000
```

```
Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2237487D+01       Std.Dev of LHS    =   .8005484D+00
StdDev of residuals=   .5657431D+00       Sum of squares    =   .2006809D+03
R-squared          =   .5130102D+00       Adjusted R-squared=   .5005830D+00
F[ 16,   627]      =   .4128133D+02
Log-likelihood     =  -.5383500D+03       Restr.(·=0) Log-l =  -.7700329D+03
Amemiya Pr. Criter.=   .1724690D+01       Akaike Info.Crit. =   .3285142D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .2114035D+03          16.              .1321272D+02
       Residual        .2006809D+03         627.              .3200652D+00
       Total           .4120844D+03         643.              .6408777D+00
Estd. Autocorrelation of e(i,t)    -.014648
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .39146        .7044E-01    5.558   .00000   .28846        .45658
ORDTAREA   .10754E-01    .1547E-01     .695   .48696   2.9132        1.4926
INTENTOV  -.42541E-01    .2789E-01   -1.525   .12724   2.4277        1.2168
TAG2       .86349        .4645E-01   18.589   .00000   .50802        .49648
CC         .68642        .9880E-01    6.948   .00000   .62500E-01    .23848



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           1.55946               .12285
                    2           1.30408               .12090
                    3           1.76191               .12437
                    4           1.45328               .12090
                    5           1.70472               .12139
                    6           1.91913               .12357
                    7           1.91140               .12556
                    8           1.58532               .12293
                    9           1.91239               .12050
                   10           1.67775               .12807
                   11           1.70352               .12130
                   12           1.26118               .12191

Test Statistics for the Classical Model

       Model              Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -770.03289          .412084D+03     .0000000
(2)  Group effects only     -749.41465          .386525D+03     .0620248
(3)  X - variables only     -582.94408          .230490D+03     .4406731
(4)  X and group effects    -538.35002          .200681D+03     .5130102

                              Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)    41.236     11     .00002          3.799   11   631     .00003
 (3) vs (1)   374.178      5     .00000        100.531    5   638     .00000
 (4) vs (1)   463.366     16     .00000         41.281   16   627     .00000
 (4) vs (2)   422.129      5     .00000        116.129    5   627     .00000
 (4) vs (3)    89.188     11     .00000          8.467   11   627     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .31453D-01      .92944D-01
(Used  Means.  Q  =     .0191)
Estimates:  Var[e]             =   .320065D+00
            Var[u]             =   .253456D-01
```

```
           Corr[v(i,t),v(i,s)] =   .073378
Lagrange Multiplier Test vs. Model (3) =     189.99106
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =      12.38671
( 5 df, prob value =  .029856)
Estd. Autocorrelation of e(i,t)    -.014088
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .319921D+00
            Var[u]              =   .909819D-01
            Sum of Squares          .231085D+03
            R-squared               .439230D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .40231        .7017E-01    5.734   .00000   .28846        .45658
ORDTAREA   .11290E-01    .1546E-01     .730   .46511   2.9132        1.4926
INTENTOV  -.34237E-01    .2754E-01   -1.243   .21386   2.4277        1.2168
TAG2       .86300        .4645E-01   18.580   .00000   .50802        .49648
CC         .68581        .9878E-01    6.943   .00000   .62500E-01    .23848
Constant   1.6260        .1078       15.078   .00000
```

Tabla 6.14: Resultados para la variable tiempo de realización sin errores utilizando sólo el modelo TAG.

En este caso, la ecuación con sólo la variable TAG explica una proporción de varianza menor (R^2=0.439) que la explicada por el modelo con todas las variables (R^2= 0.711). El contraste entre ambos modelos produce una $F_{2,635}$=147.92, p<0.01, dando a entender que al incluir las variables correspondientes al análisis NGOMS aumenta la proporción de varianza explicada.

Un resultado importante que es posible observar en la tabla 6.14 es que, cuando se considera TAG de modo independiente, su relación con el tiempo para realizar la tarea es positiva, y no negativa como aparecía en la tabla 6.12.

Resultados para la variable dependiente tiempo con errores

A continuación se muestran los resultados para la variable tiempo con error.

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between       39.4712              11.         3.58829
          Residual      580.087              632.         .917859
          Total         619.558              643.         .963543


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =           644       Weights           =  ONE
Mean of LHS        =   .2233764D+01       Std.Dev of LHS    =   .9816021D+00
StdDev of residuals=   .6501093D+00       Sum of squares    =   .2688004D+03
R-squared          =   .5661416D+00       Adjusted R-squared=   .5613664D+00
F[  7,   636]      =   .1185594D+03
Log-likelihood     =  -.6324555D+03       Restr.(·=0) Log-l =  -.9013374D+03
Amemiya Pr. Criter.=   .1988992D+01       Akaike Info.Crit. =   .4278923D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .3507575D+03            7.              .5010821D+02
       Residual        .2688004D+03          636.              .4226421D+00
       Total           .6195579D+03          643.              .9635426D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .64518        .8385E-01    7.694   .00000   .26392        .43597
ORDTAREA  -.15488        .2045E-01   -7.575   .00000   2.6656        1.3765
```

```
INTENTOV  -.81017E-02    .3205E-01    -.253   .80041   2.2213        1.1445
NGOMS      .32432E-01    .3094E-02   10.482   .00000   5.5980        9.3314
TAG2      -.10433        .8900E-01   -1.172   .24111   .46480        .50826
TGOMS      .82595E-01    .6243E-02   13.229   .00000   18.302        7.4339
CC         .43711E-01    .1212         .361   .71839   .57184E-01    .24054
Constant   .90029        .1287        6.994   .00000


Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2233764D+01       Std.Dev of LHS    =   .9816021D+00
StdDev of residuals=   .5902256D+00       Sum of squares    =   .2177289D+03
R-squared          =   .6485737D+00       Adjusted R-squared=   .6384527D+00
F[ 18,   625]      =   .6408150D+02
Log-likelihood     =  -.5646042D+03       Restr.(·=0) Log-l =  -.9013374D+03
Amemiya Pr. Criter.=   .1812435D+01       Akaike Info.Crit. =   .3586441D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .4018290D+03          18.              .2232383D+02
       Residual        .2177289D+03         625.              .3483663D+00
       Total           .6195579D+03         643.              .9635426D+00
Estd. Autocorrelation of e(i,t)    -.004101
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .50317        .7765E-01    6.480   .00000   .26392        .43597
ORDTAREA  -.16011        .1869E-01   -8.568   .00000   2.6656        1.3765
INTENTOV  -.11270        .3112E-01   -3.621   .00029   2.2213        1.1445
NGOMS      .32513E-01    .2811E-02   11.567   .00000   5.5980        9.3314
TAG2      -.10675        .8108E-01   -1.316   .18801   .46480        .50826
TGOMS      .82897E-01    .5688E-02   14.575   .00000   18.302        7.4339
CC         .37630E-01    .1101         .342   .73254   .57184E-01    .24054



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           1.04622               .15184
                    2            .72933               .15065
                    3           1.52021               .15239
                    4            .75109               .15065
                    5           1.13492               .15150
                    6           1.35460               .15277
                    7           1.52223               .15635
                    8            .99396               .15181
                    9           1.43471               .14959
                   10           1.56304               .15596
                   11           1.13084               .15084
                   12            .70274               .15034

Test Statistics for the Classical Model

       Model               Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -901.33741          .619558D+03     .0000000
(2)  Group effects only     -880.14059          .580087D+03     .0637087
(3)  X - variables only     -632.45547          .268800D+03     .5661416
(4)  X and group effects    -564.60417          .217729D+03     .6485737

                            Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)    42.394     11     .00001          3.909   11   631     .00002
 (3) vs (1)   537.764      7     .00000        118.559    7   636     .00000
 (4) vs (1)   673.466     18     .00000         64.082   18   625     .00000
```

```
 (4) vs (2)   631.073      7     .00000        148.595    7   625     .00000
 (4) vs (3)   135.703     11     .00000         13.328   11   625     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .21787D+00      .21787D+00
(Used    OLS.  Q  =     .0191)
Estimates:  Var[e]              =   .348366D+00
            Var[u]              =   .211219D+00
            Corr[v(i,t),v(i,s)] =   .377456
Lagrange Multiplier Test vs. Model (3) =     443.76968
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =        .00010
( 7 df, prob value = 1.000000)
Estd. Autocorrelation of e(i,t)    -.003877
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .348566D+00
            Var[u]              =   .248766D+00
            Sum of Squares          .274385D+03
            R-squared               .557128D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .50691        .7760E-01    6.532   .00000   .26392        .43597
ORDTAREA  -.15987        .1868E-01   -8.557   .00000   2.6656        1.3765
INTENTOV  -.10953        .3106E-01   -3.526   .00042   2.2213        1.1445
NGOMS      .32507E-01    .2811E-02   11.565   .00000   5.5980        9.3314
TAG2      -.10777        .8108E-01   -1.329   .18379   .46480        .50826
TGOMS      .82960E-01    .5688E-02   14.586   .00000   18.302        7.4339
CC         .37584E-01    .1101         .341   .73285   .57184E-01    .24054
Constant   1.1474        .1898        6.047   .00000
```

Tabla 6.15: Resultados para la variable tiempo de realización con errores.

Los resultados favorecen el modelo de efectos fijos (Hausman=84; p<.001). Todos los coeficientes aparecen como significativos. salvo TAG. La proporción de varianza explicada es de 0.60.

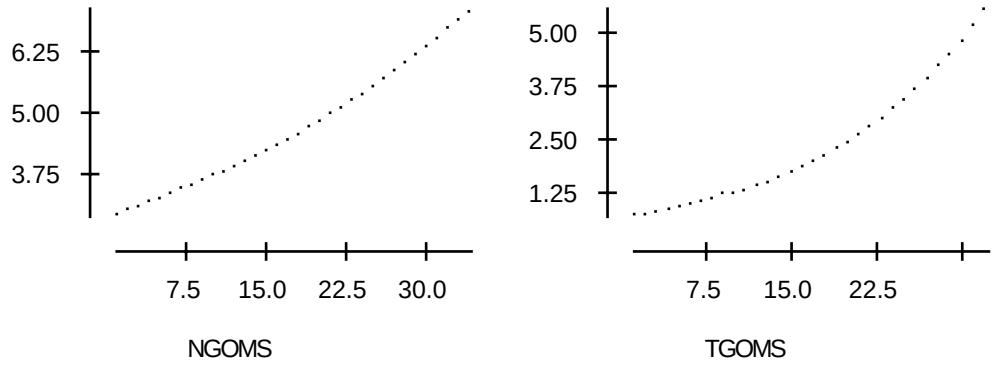


Figura 6.19: Influencia de NGOMS y TGOMS asumiendo valores medios en el resto de las variables independientes en la escala normal.

Todas las variables presentan coeficientes altos con un nivel de significación menor del 1%. salvo TAG Al parecer, los modelos trabajados son capaces de afrontar la situación en la que se intenta predecir el tiempo que los sujetos emplearán en equivocarse y en recuperarse de los errores cometidos. Cada regla nueva significa aproximadamente 0.85 s. de tiempo.

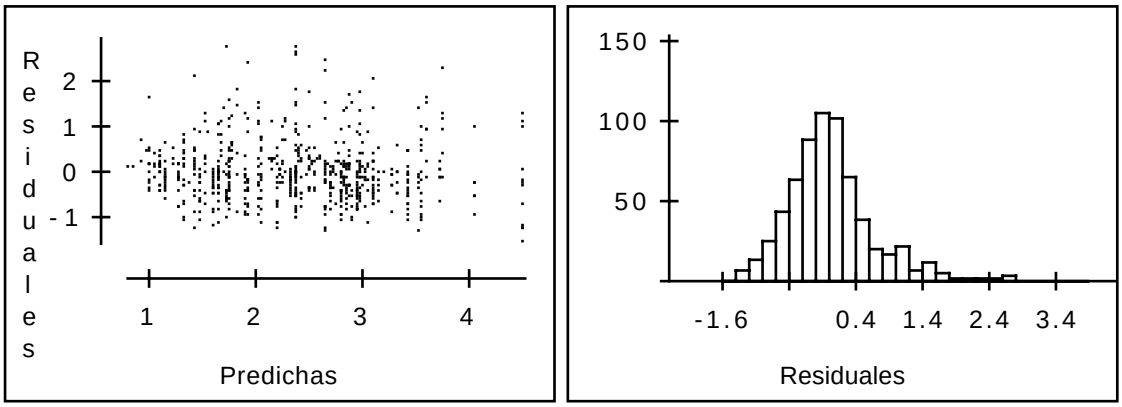


Figura 6.20: Análisis de residuales para la variable dependiente tiempo con error.

Los gráficos de residuales muestran un comportamiento correcto de éstos.

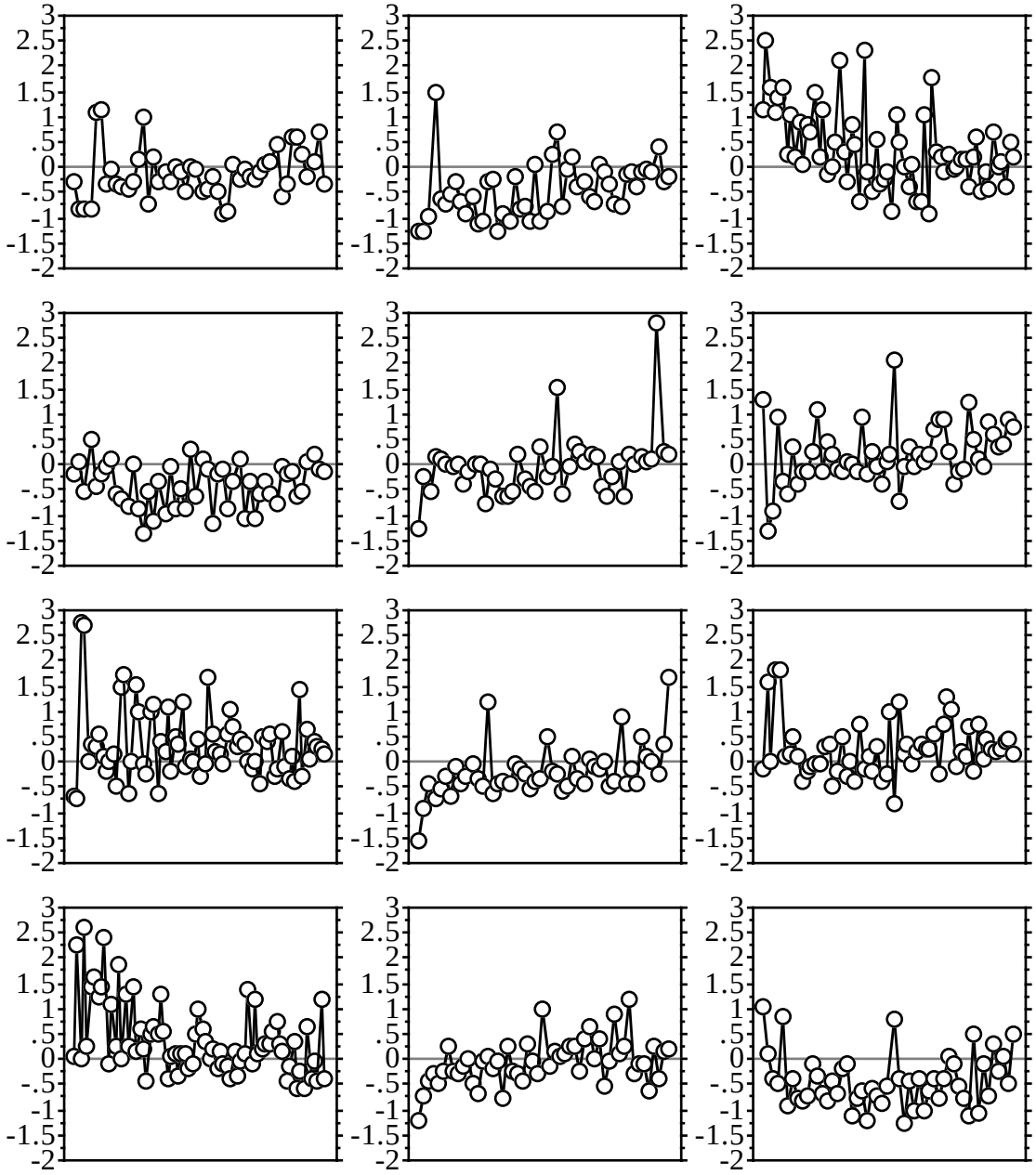


Figura 6.21: Gráficos de residuales por series temporales para la variable tiempo con error.

Resultados no mostrados aquí revelaron como la transformación realizada eliminó la mayor parte de la autocorrelación intrasujeto para todos los individuos hasta el nivel 14.

La variable tiempo con error incluye el tiempo que los usuarios emplearon en averiguar cuál era el procedimiento correcto para continuar la tarea. Una situación esperable en este caso sería que los usuarios se encontraran en muchas ocasiones empleando tiempos muy considerables con problemas concretos, generando de este modo residuales de un tamaño inusitadamente grande. Comparando los gráficos de la figura 21 con los de la figura 15 es posible apreciar visualmente este efecto.

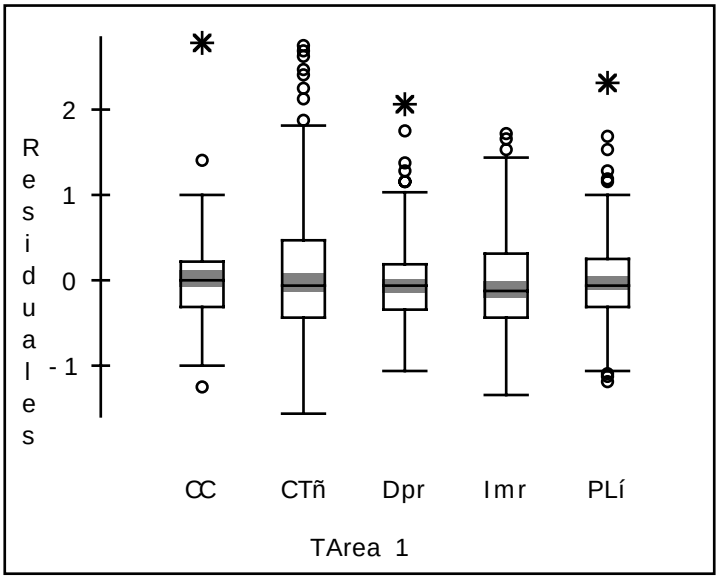


Figura 6.22: Gráfico de cajas para la tarea 1.

En el gráfico de cajas y bigotes aparecen como es de esperar mayor número de valores extremos que el equivalente para la variable sin errores (gráfico 5.16). El análisis de varianza mostrado en la tabla 6.15 no muestra diferencias globales entre las diversas tareas.

Fuente	gl	SC	MC	F	Prob
Const	1	0.593643	0.593643	1.3309	0.2491
Tarea 1	4	2.44059	0.610146	1.3679	0.2435
Error	651	290.378	0.446049		
Total	655	292.819			

Tabla 6.15: Análisis de varianza para los residuales.

Comparación entre modelos

A continuación se muestran los resultados sólo utilizando las variables provenientes de los análisis NGOMS o del análisis TAG en cada caso. De este

modo es posible evaluar la capacidad de predicción de cada uno de los modelos por separado.

Resultados para la variable dependiente tiempo con errores utilizando NGOMS

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between       39.4712              11.         3.58829
          Residual      580.087             632.         .917859
          Total         619.558             643.         .963543


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =           644       Weights           =  ONE
Mean of LHS        =   .2233764D+01       Std.Dev of LHS    =   .9816021D+00
StdDev of residuals=   .6503002D+00       Sum of squares    =   .2693811D+03
R-squared          =   .5652043D+00       Adjusted R-squared=   .5611089D+00
F[  6,   637]      =   .1380093D+03
Log-likelihood     =  -.6331504D+03       Restr.(·=0) Log-l =  -.9013374D+03
Amemiya Pr. Criter.=   .1988045D+01       Akaike Info.Crit. =   .4274869D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .3501768D+03           6.              .5836279D+02
       Residual        .2693811D+03         637.              .4228903D+00
       Total           .6195579D+03         643.              .9635426D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
-------------------------------------------------------------------------------
PRINTEN    .64445        .8387E-01    7.684   .00000   .26392        .43597
ORDTAREA  -.14746        .1945E-01   -7.583   .00000   2.6656        1.3765
INTENTOV  -.83501E-02    .3205E-01    -.260   .79448   2.2213        1.1445
NGOMS      .31137E-01    .2891E-02   10.771   .00000   5.5980        9.3314
TGOMS      .76907E-01    .3929E-02   19.572   .00000   18.302        7.4339
CC         .61598E-01    .1203         .512   .60859   .57184E-01    .24054
Constant   .94579        .1235        7.660   .00000


Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =           644       Weights           =  ONE
Mean of LHS        =   .2233764D+01       Std.Dev of LHS    =   .9816021D+00
StdDev of residuals=   .5905711D+00       Sum of squares    =   .2183327D+03
R-squared          =   .6475992D+00       Adjusted R-squared=   .6380292D+00
F[ 17,   626]      =   .6766980D+02
Log-likelihood     =  -.5654959D+03       Restr.(·=0) Log-l =  -.9013374D+03
Amemiya Pr. Criter.=   .1812099D+01       Akaike Info.Crit. =   .3585226D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .4012252D+03          17.              .2360148D+02
       Residual        .2183327D+03         626.              .3487742D+00
       Total           .6195579D+03         643.              .9635426D+00
Estd. Autocorrelation of e(i,t)    -.002778
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
-------------------------------------------------------------------------------
PRINTEN    .50251        .7769E-01    6.468   .00000   .26392        .43597
ORDTAREA  -.15246        .1777E-01   -8.579   .00000   2.6656        1.3765
INTENTOV  -.11288        .3114E-01   -3.625   .00029   2.2213        1.1445
NGOMS      .31186E-01    .2626E-02   11.878   .00000   5.5980        9.3314
TGOMS      .77066E-01    .3571E-02   21.584   .00000   18.302        7.4339
CC         .55934E-01    .1093         .512   .60881   .57184E-01    .24054



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           1.09161              .14796
                    2            .77719              .14628
                    3           1.56349              .14889
```

```
                    4           .79894              .14628
                    5          1.18334              .14705
                    6          1.40109              .14872
                    7          1.57466              .15128
                    8          1.03936              .14793
                    9          1.48064              .14555
                   10          1.60600              .15259
                   11          1.17787              .14663
                   12           .74680              .14665

Test Statistics for the Classical Model

       Model              Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -901.33741          .619558D+03     .0000000
(2)  Group effects only     -880.14059          .580087D+03     .0637087
(3)  X - variables only     -633.15039          .269381D+03     .5652043
(4)  X and group effects    -565.49585          .218333D+03     .6475992

                              Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)    42.394     11     .00001          3.909   11   631     .00002
 (3) vs (1)   536.374      6     .00000        138.009    6   637     .00000
 (4) vs (1)   671.683     17     .00000         67.670   17   626     .00000
 (4) vs (2)   629.289      6     .00000        172.869    6   626     .00000
 (4) vs (3)   135.309     11     .00000         13.306   11   626     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .10771D-01      .17422D+00
(Used  Means.  Q  =     .0191)
Estimates:  Var[e]              =   .348774D+00
            Var[u]              =   .411586D-02
            Corr[v(i,t),v(i,s)] =   .011663
Lagrange Multiplier Test vs. Model (3) =     441.31895
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =      95.87299
( 6 df, prob value =  .000000)
Estd. Autocorrelation of e(i,t)     .006296
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .351559D+00
            Var[u]              =   .178785D+00
            Sum of Squares          .270367D+03
            R-squared               .563614D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .58975        .7676E-01    7.683   .00000   .26392        .43597
ORDTAREA  -.14912        .1770E-01   -8.424   .00000   2.6656        1.3765
INTENTOV  -.48151E-01    .2991E-01   -1.610   .10747   2.2213        1.1445
NGOMS      .31138E-01    .2625E-02   11.861   .00000   5.5980        9.3314
TGOMS      .76978E-01    .3569E-02   21.566   .00000   18.302        7.4339
CC         .59112E-01    .1093         .541   .58849   .57184E-01    .24054
Constant   1.0432        .1219        8.554   .00000
```

Tabla 6.16: Resultados para la variable tiempo de realización con errores.

Como es posible ver, la ecuación que incluye sólo las variables extraidas del análisis NGOMS (R^2= 0.6475) explican la misma cantidad de varianza que la ecuación introduciendo la ecuación con las variables de ambos modelos (R^2= 0.6485).

Resultados para la variable dependiente tiempo con errores utilizando TAG.

```
          Unconditional ANOVA (No regressors)
          Source          Variation        Deg. Free.     Mean Square
          Between       39.4712              11.         3.58829
          Residual      580.087             632.         .917859
          Total         619.558             643.         .963543


OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2233764D+01       Std.Dev of LHS    =   .9816021D+00
StdDev of residuals=   .7622825D+00       Sum of squares    =   .3707256D+03
R-squared          =   .4016288D+00       Adjusted R-squared=   .3969393D+00
F[  5,   638]      =   .8564554D+02
Log-likelihood     =  -.7359763D+03       Restr.(·=0) Log-l =  -.9013374D+03
Amemiya Pr. Criter.=   .2304274D+01       Akaike Info.Crit. =   .5864884D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .2488323D+03           5.              .4976645D+02
       Residual        .3707256D+03         638.              .5810747D+00
       Total           .6195579D+03         643.              .9635426D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .62159        .9826E-01    6.326   .00000   .26392        .43597
ORDTAREA  -.47358E-01    .2240E-01   -2.114   .03449   2.6656        1.3765
INTENTOV  -.12498E-01    .3757E-01    -.333   .73942   2.2214        1.1444
TAG2       .96606        .6173E-01  15.649   .00000   .46480        .50826
CC         .70846        .1328       5.334   .00000   .57184E-01    .24054
Constant   1.8430        .1300      14.175   .00000


Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =            644       Weights           =  ONE
Mean of LHS        =   .2233764D+01       Std.Dev of LHS    =   .9816021D+00
StdDev of residuals=   .7139995D+00       Sum of squares    =   .3196416D+03
R-squared          =   .4840811D+00       Adjusted R-squared=   .4709157D+00
F[ 16,   627]      =   .3676921D+02
Log-likelihood     =  -.6882360D+03       Restr.(·=0) Log-l =  -.9013374D+03
Amemiya Pr. Criter.=   .2190174D+01       Akaike Info.Crit. =   .5232525D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .2999163D+03          16.              .1874477D+02
       Residual        .3196416D+03         627.              .5097952D+00
       Total           .6195579D+03         643.              .9635426D+00
Estd. Autocorrelation of e(i,t)    -.020884
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRINTEN    .48200        .9388E-01    5.134   .00000   .26392        .43597
ORDTAREA  -.51280E-01    .2110E-01   -2.430   .01511   2.6656        1.3765
INTENTOV  -.11530        .3765E-01   -3.063   .00219   2.2214        1.1444
TAG2       .96883        .5785E-01  16.747   .00000   .46480        .50826
CC         .70198        .1245       5.638   .00000   .57184E-01    .24054



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           1.96865              .16772
                    2           1.68672              .16496
                    3           2.41607              .16925
                    4           1.70848              .16496
                    5           2.10720              .16560
                    6           2.28486              .16866
                    7           2.52935              .17086
                    8           1.91645              .16767
                    9           2.37698              .16413
```

```
                    10          2.45585              .17410
                    11          2.08417              .16541
                    12          1.61056              .16624

Test Statistics for the Classical Model

      Model              Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -901.33741          .619558D+03     .0000000
(2)  Group effects only     -880.14059          .580087D+03     .0637087
(3)  X - variables only     -735.97627          .370726D+03     .4016288
(4)  X and group effects    -688.23595          .319642D+03     .4840811

                              Hypothesis Tests

             Likelihood Ratio Test               F Tests
             Chi-squared   d.f.  Prob value      F    num. denom.  Prob value

 (2) vs (1)    42.394     11     .00001          3.909   11   631     .00002
 (3) vs (1)   330.722      5     .00000         85.646    5   638     .00000
 (4) vs (1)   426.203     16     .00000         36.769   16   627     .00000
 (4) vs (2)   383.809      5     .00000        102.176    5   627     .00000
 (4) vs (3)    95.481     11     .00000          9.110   11   627     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .31102D-01      .14549D+00
(Used  Means.  Q  =     .0191)
Estimates:  Var[e]              =   .509795D+00
            Var[u]              =   .213742D-01
            Corr[v(i,t),v(i,s)] =   .040240
Lagrange Multiplier Test vs. Model (3) =     215.21233
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =      27.88537
( 5 df, prob value =  .000038)
Estd. Autocorrelation of e(i,t)    -.017328
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .510796D+00
            Var[u]              =   .152475D+00
            Sum of Squares          .373593D+03
            R-squared               .397001D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRINTEN    .52522        .9332E-01    5.628   .00000   .26392        .43597
ORDTAREA  -.49812E-01    .2106E-01   -2.365   .01804   2.6656        1.3765
INTENTOV  -.83010E-01    .3691E-01   -2.249   .02451   2.2214        1.1444
TAG2       .96774        .5784E-01   16.730   .00000   .46480        .50826
CC         .70373        .1245        5.654   .00000   .57184E-01    .24054
Constant   2.0156        .1397       14.426   .00000
```

Tabla 6.17: Resultados para la variable tiempo de realización con errores utilizando sólo el modelo TAG.

En este caso, la ecuación utilizando sólo la variable extraída de TAG explica una proporción de varianza menor (R^2=0.484) que la explicada por el modelo con todas las variables (R^2= 0.6485). El contraste entre ambos modelos produce una $F_{2,635}$=298.82, p<0.01, dando a entender que al incluir las variables correspondientes al análisis NGOMS aumenta la proporción de varianza explicada.

3. Resultados para el Nivel 3

En el nivel 3 se manejan 1971 observaciones para las distintas estimaciones de complejidad de las situaciones realizadas por el método de Action-Task

Debido a que las acciones consideradas aquí son de una duración menor, varios valores recibieron un tiempo igual a cero en las variables originales (sin transformar). Ya que el logaritmo de estos valores no existe, la transformación realizada eliminaba los casos de los análisis. Debido al poco número de ocasiones en que esto ocurrió (23), no se tomó ninguna medida en especial para modificar las variables.

Los histogramas de las variables dependientes y sus logaritmos (en base decimal) son mostrados a continuación.

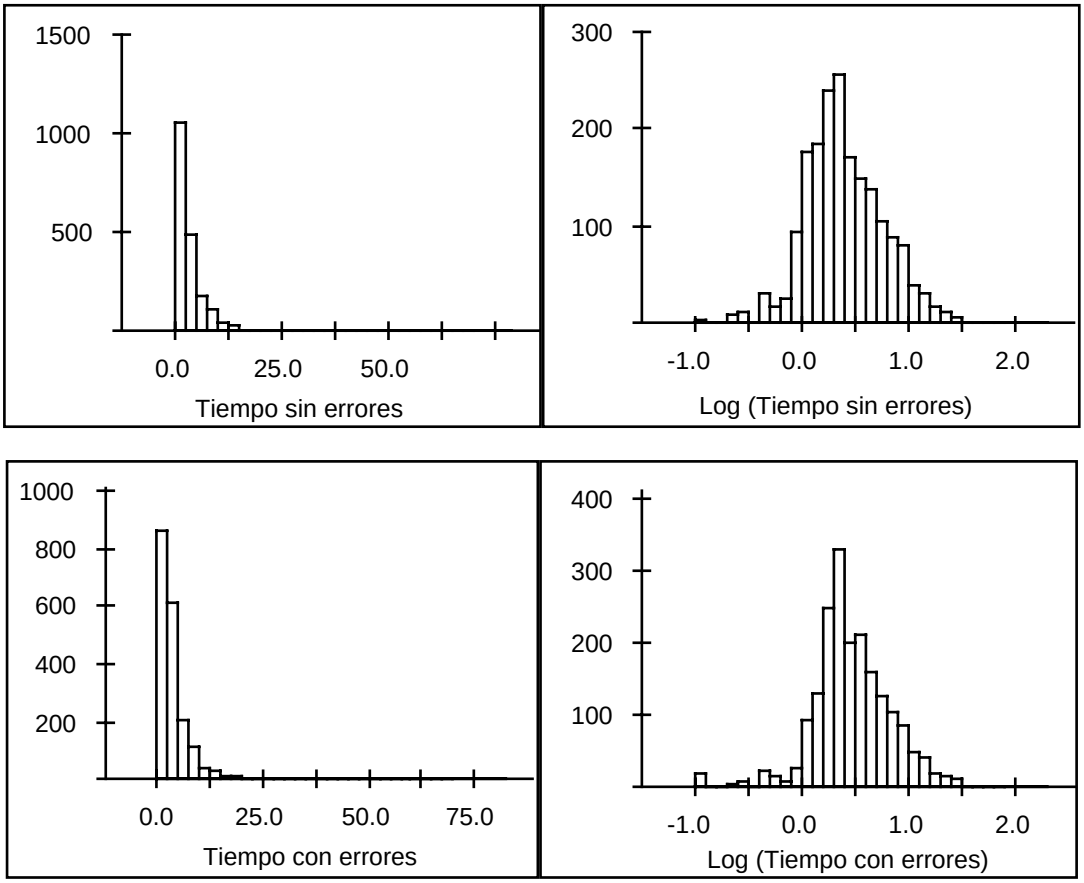


Figura 6.23: Histogramas de las variables dependientes consideradas.

	TSE (s)	Log(TSE)	TCE (s)	Log(TCE)
Medias	3.75	0.39	4.18	0.45
Mediana	2.30	0.36	2.70	0.43
N	1971	1971	1971	1971

Asimet.	5.03	0.096	4.89	-0.38
Curt.	43.58	0.71	41.54	2.15
Desv.	4.69	0.39	4.72	0.37
Interc.	3	0.49	3.10	0.43
Mínimo	0	-1	0.10	-1
Máximo	68.60	1.83	68.70	1.83

Tabla 6.18: Estadísticos descriptivos de las variables dependientes consideradas.

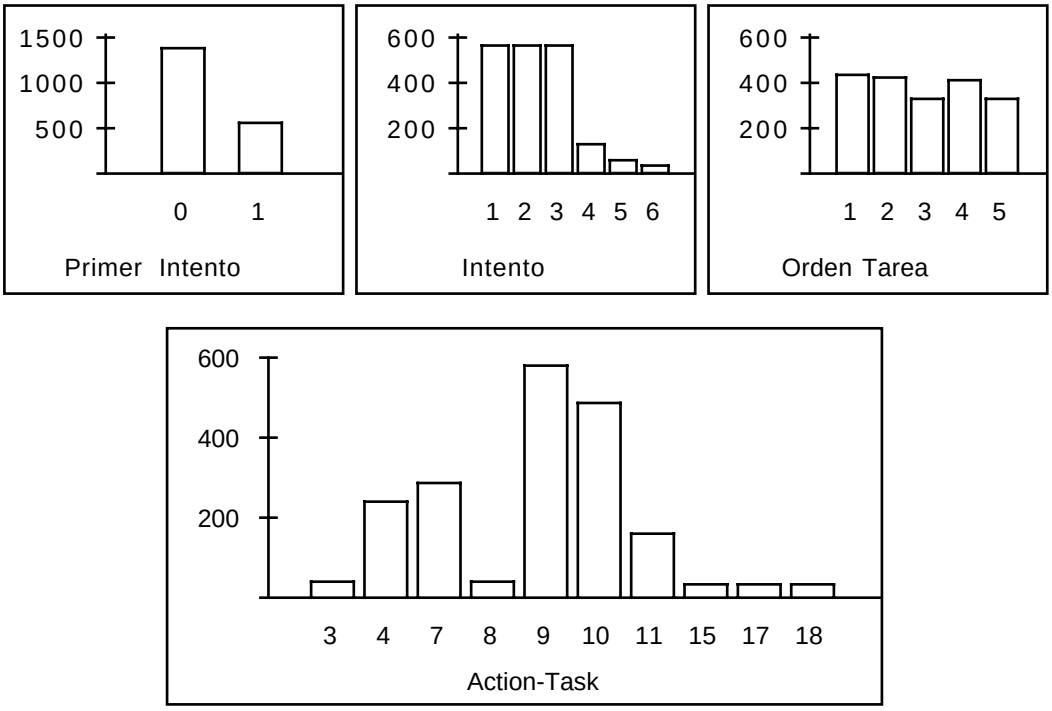


Figura 6.24: Diagramas de barras de las variables independientes.

Orden tarea	Primer intento	Intento	Acción
1.000			
0.025	1.000		
-0.086	-0.719	1.000	
-0.050	0.004	-0.016	1.000

Tabla 6.19: Correlaciones entre las variables independientes utilizadas.

No existen correlaciones muy altas salvo la de Intento con el Primer intento. *Action-Task* está poco relacionada con las otras variables.

Resultados para la variable dependiente tiempo sin errores

A continuación se muestran los resultados para la variable dependiente tiempo sin errores.

```
OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =          1928       Weights           =  ONE
Mean of LHS        =   .4123027D+00       Std.Dev of LHS    =   .3865854D+00
StdDev of residuals=   .3684854D+00       Sum of squares    =   .2611078D+03
R-squared          =   .9333402D-01       Adjusted R-squared=   .9144808D-01
F[  4,  1923]      =   .4948937D+02
Log-likelihood     =  -.8083835D+03       Restr.(·=0) Log-l =  -.9028373D+03
Amemiya Pr. Criter.=   .8437588D+00       Akaike Info.Crit. =   .1361336D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .2687897D+02           4.               .6719741D+01
       Residual        .2611078D+03         1923.              .1357815D+00
       Total           .2879868D+03         1927.              .1494483D+00
Variable  Coefficient  Std. Error   t-ratio Prob t ≥x  Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRIMERINTEN.19877        .2537E-01    7.835   .00000   .30630        .47463
ORDTAREA  -.13613E-01    .5706E-02   -2.385   .01706   3.0454        1.4819
INTENTO    .78470E-02    .9769E-02     .803   .42182   2.4449        1.2379
ACCION     .25382E-01    .2824E-02    8.987   .00000   9.2945        2.9779
Constant   .13121        .4628E-01    2.835   .00458



Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGSIN
Observations       =          1928       Weights           =  ONE
Mean of LHS        =   .4123027D+00       Std.Dev of LHS    =   .3865854D+00
StdDev of residuals=   .3560803D+00       Sum of squares    =   .2424285D+03
R-squared          =   .1581956D+00       Adjusted R-squared=   .1515915D+00
F[ 15,  1912]      =   .2395411D+02
Log-likelihood     =  -.7368292D+03       Restr.(·=0) Log-l =  -.9028373D+03
Amemiya Pr. Criter.=   .7809431D+00       Akaike Info.Crit. =   .1278454D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .4555826D+02          15.               .3037217D+01
       Residual        .2424285D+03         1912.              .1267932D+00
       Total           .2879868D+03         1927.              .1494483D+00
Estd. Autocorrelation of e(i,t)    -.010883
Variable  Coefficient  Std. Error   t-ratio Prob t ≥x  Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRIMERINTEN .17349       .2502E-01    6.934   .00000   .30630        .47463
ORDTAREA   -.13130E-01    .5534E-02   -2.373   .01767   3.0454        1.4819
INTENTO    -.12424E-01    .1012E-01   -1.228   .21951   2.4449        1.2379
ACCION      .25296E-01    .2729E-02    9.268   .00000   9.2945        2.9779



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1            .12558              .05077
                    2            .03805              .05066
                    3            .23423              .05115
                    4            .04545              .05017
                    5            .20435              .05032
                    6            .28705              .04990
                    7            .27901              .05198
                    8            .16046              .05079
                    9            .32115              .05024
                   10            .20676              .05248
                   11            .19289              .05030
                   12            .01684              .05080

Test Statistics for the Classical Model

       Model              Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -902.83732         .287987D+03     .0000000
```

```
(2)  Group effects only     -844.17279          .270984D+03     .0590406
(3)  X - variables only     -808.38348          .261108D+03     .0933340
(4)  X and group effects    -736.82915          .242429D+03     .1581956

                              Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)   117.329     11     .00000         10.929   11  1915     .00000
 (3) vs (1)   188.908      4     .00000         49.489    4  1923     .00000
 (4) vs (1)   332.016     15     .00000         23.954   15  1912     .00000
 (4) vs (2)   214.687      4     .00000         56.303    4  1912     .00000
 (4) vs (3)   143.109     11     .00000         13.393   11  1912     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
Lagrange Multiplier Test vs. Model (3) =     557.17197
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =       7.63597
( 4 df, prob value =  .105861)
Estd. Autocorrelation of e(i,t)    -.010835
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .126797D+00
            Var[u]              =   .171216D-01
            Sum of Squares          .261776D+03
            R-squared               .910133D-01
Variable  Coefficient  Std. Error   t-ratio Prob t ≥x   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRIMERINTEN .17632        .2497E-01    7.061   .00000   .30630        .47463
ORDTAREA   -.13168E-01    .5533E-02   -2.380   .01731   3.0454        1.4819
INTENTO    -.10174E-01    .1005E-01   -1.012   .31146   2.4449        1.2379
ACCION      .25303E-01    .2729E-02    9.271   .00000   9.2945        2.9779
Constant    .17097        .4920E-01    3.475   .00051
```

Tabla 6.20: Resultados para la variable tiempo sin errores.

Los resultados favorecen el modelo de efectos aleatorios (Hausman=7.63, p=.1058) frente al de efectos fijos. En este caso, la proporción de varianza explicada es de R^2=.0.09. El coeficiente de la variable acción resulta significativo (t=9.268; p<.01) con un coeficiente de 0.025. El siguiente gráfico muestra el crecimiento en las puntuaciones predichas en la variable sin transformar como resultado del crecimiento en la variable Action-Task manteniendo el resto de las variables en nivel medio. Un efecto de aproximadamente 0.16 s. puede esperarse del aumento en una acción en el tiempo de realización de la tarea.

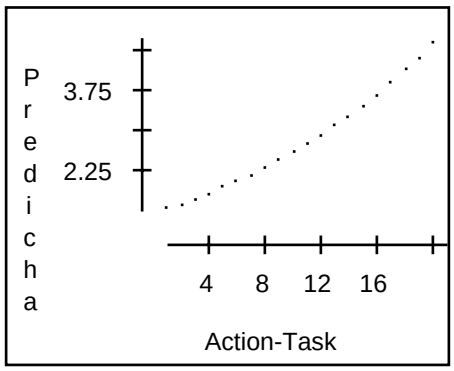


Figura 6.25: Influencia Action asumiendo valores medios en el resto de las variables independientes.

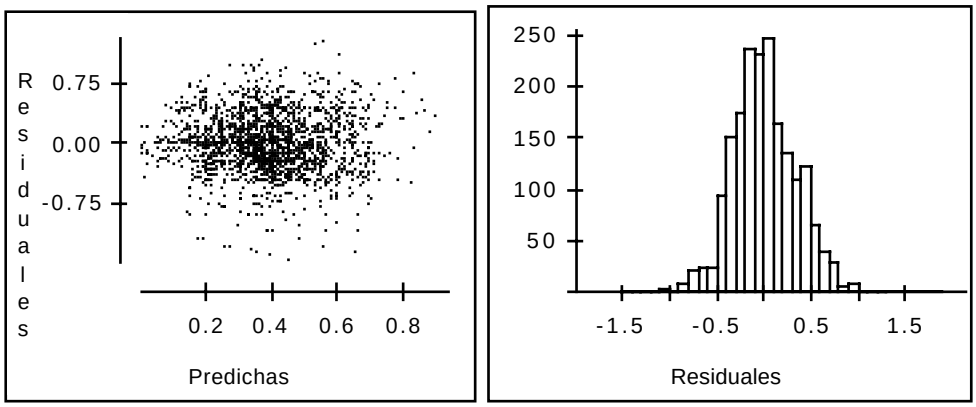


Figura 6.26: Graficos de residuales para la variable dependiente sin error.

Los gráficos de residuales muestran una apariencia adecuada.

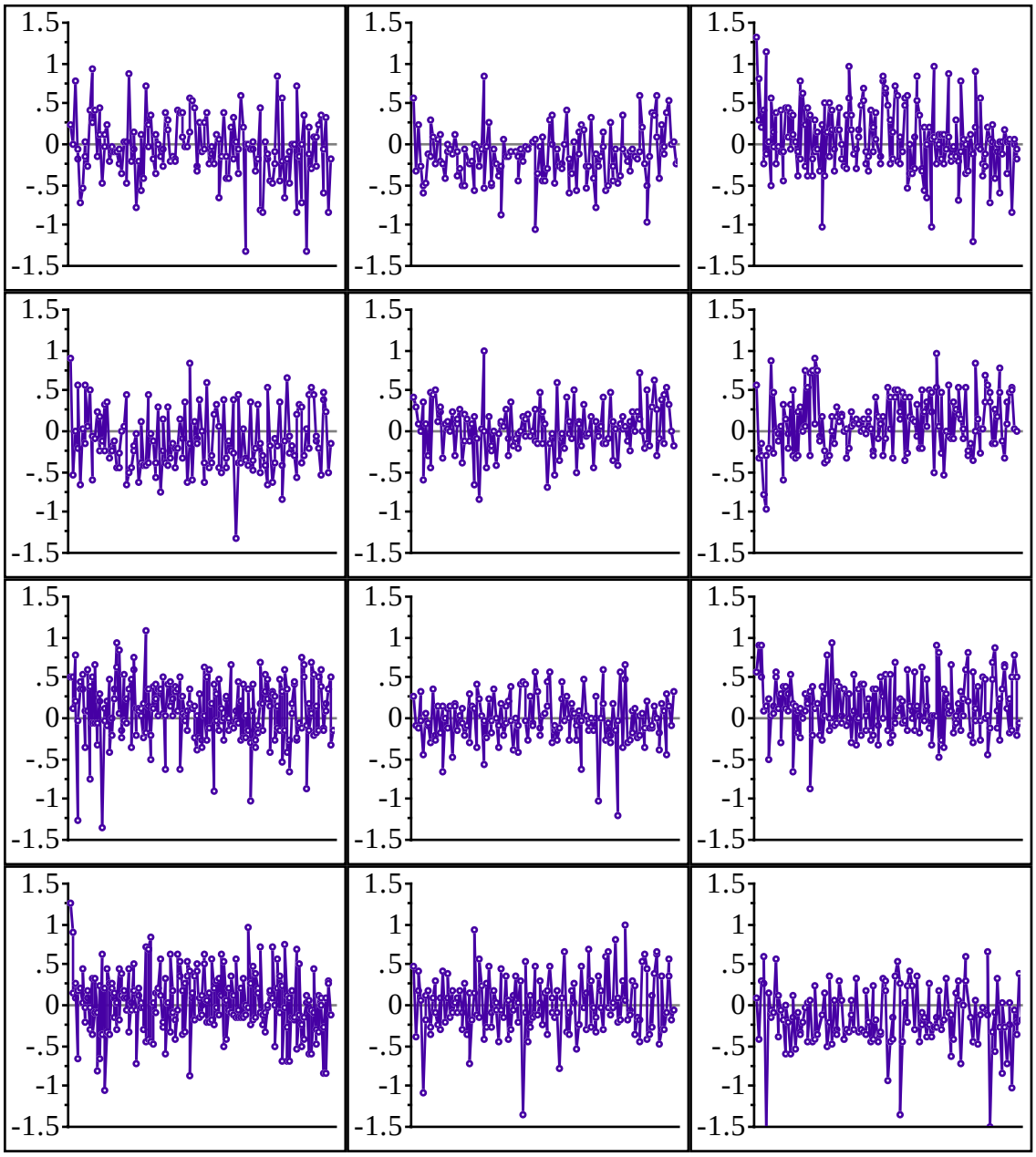


Figura 6.27: Gráficos de residuales por series temporales para la variable tiempo sin error.

La autocorrelación intrasujeto para todos los individuos hasta el nivel 14 en este caso desapareció también prácticamente del todo gracias a las correciones realizadas.

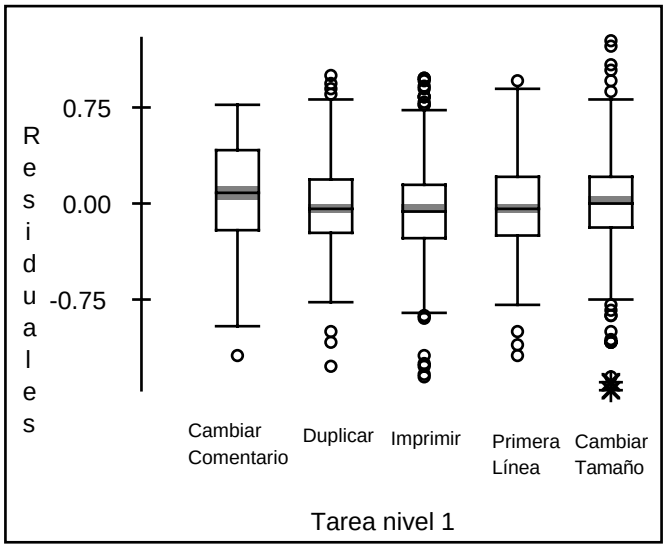


Figura 6.28: Gráfico de cajas para la tarea 1.

Fuente	gl	SC	MC	F	Prob
Const	1	0.020379	0.020379	0.16097	0.6883
Tn1	4	4.14107	1.03527	8.1771	≤ 0.0001
Error	1935	244.982	0.126606		
Total	1939	249.123			

Tabla 6.21: Análisis de varianza para los residuales.

Al parecer existen diferencias entre las residuales según las tareas. Las pruebas a posteriori indican que estas diferencias parecen centrarse fundamentalmente en Cambiar Comentario, la cual suele tener residuales positivos.

	Diferencia	err. típ.	Prob.
Guardar Como - Comentario	-0.104906	0.0278	0.001624
Imprimir - Comentario	-0.154364	0.0272	0.000000
Imprimir - Guardar Como	-0.049458	0.0240	0.329191
Prim. linea - Comentario	-0.104823	0.0288	0.002860
Prim. linea - Guardar Como	0.000083	0.0259	1
Prim. linea - Imprimir	0.049541	0.0252	0.398187
Tipo tamaño - Comentario	-0.091553	0.0275	0.008883
Tipo tamaño - Guardar Como	0.013353	0.0244	0.999843
Tipo tamaño - Imprimir	0.062811	0.0237	0.077429

Tipo tamaño - Prim. linea	0.013270	0.0256	0.999905

Tabla 6.22: Pruebas a posteriori para los residuales.

Resultados para las variable dependiente tiempo con errores

A continuación se muestran los resultados para la variable tiempo con errores.

```
OLS Without Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =          1950       Weights           =  ONE
Mean of LHS        =   .4743377D+00       Std.Dev of LHS    =   .3767222D+00
StdDev of residuals=   .3642822D+00       Sum of squares    =   .2581044D+03
R-squared          =   .6687211D-01       Adjusted R-squared=   .6495308D-01
F[  4,  1945]      =   .3484684D+02
Log-likelihood     =  -.7952653D+03       Restr.(·=0) Log-l =  -.8627480D+03
Amemiya Pr. Criter.=   .8207850D+00       Akaike Info.Crit. =   .1330418D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .1849691D+02           4.               .4624229D+01
       Residual        .2581044D+03         1945.              .1327015D+00
       Total           .2766014D+03         1949.              .1419196D+00
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRIMERINTEN.18356        .2523E-01    7.275   .00000   .29979        .46851
ORDTAREA  -.15774E-01    .5671E-02   -2.782   .00541   3.0117        1.4662
INTENTO     .11827E-04    .9697E-02     .001   .99903   2.4228        1.2244
ACCION     .13363E-01    .2796E-02    4.779   .00000   9.1824        2.9569
Constant   .33127        .4540E-01    7.297   .00000


Least Squares with Group Dummy Variables
Ordinary    least squares regression.     Dep. Variable     =  LOGCON
Observations       =          1950       Weights           =  ONE
Mean of LHS        =   .4743377D+00       Std.Dev of LHS    =   .3767222D+00
StdDev of residuals=   .3503588D+00       Sum of squares    =   .2374009D+03
R-squared          =   .1417218D+00       Adjusted R-squared=   .1350650D+00
F[ 15,  1934]      =   .2128990D+02
Log-likelihood     =  -.7137418D+03       Restr.(·=0) Log-l =  -.8627480D+03
Amemiya Pr. Criter.=   .7484531D+00       Akaike Info.Crit. =   .1237584D+00
ANOVA  Source         Variation     Degrees of Freedom       Mean Square
       Regression      .3920043D+02          15.               .2613362D+01
       Residual        .2374009D+03         1934.              .1227513D+00
       Total           .2766014D+03         1949.              .1419196D+00
Estd. Autocorrelation of e(i,t)    -.014358
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
--------------------------------------------------------------------------------
PRIMERINTEN.15774        .2476E-01    6.371   .00000   .29979        .46851
ORDTAREA  -.14991E-01    .5474E-02   -2.739   .00617   3.0117        1.4662
INTENTO    -.19780E-01    .9992E-02   -1.980   .04774   2.4228        1.2244
ACCION     .13257E-01    .2689E-02    4.929   .00000   9.1824        2.9569



Estimated Fixed Effects

                Group       Coefficient       Standard Error
                    1           .34142              .05022
                    2           .20523              .04962
                    3           .43128              .05051
                    4           .28360              .04962
                    5           .41104              .04977
                    6           .49415              .04935
                    7           .47704              .05137
                    8           .36428              .05023
                    9           .50819              .04968
```

```
                    10            .40974              .05184
                    11            .40655              .04975
                    12            .17985              .04986

Test Statistics for the Classical Model

       Model              Log-Likelihood    Sum of Squares    R-squared

(1)  Constant term only     -862.74800          .276601D+03     .0000000
(2)  Group effects only     -793.20753          .257560D+03     .0688394
(3)  X - variables only     -795.26532          .258104D+03     .0668721
(4)  X and group effects    -713.74173          .237401D+03     .1417218

                              Hypothesis Tests

             Likelihood Ratio Test                F Tests
             Chi-squared   d.f.  Prob value       F    num. denom.  Prob value

 (2) vs (1)   139.081     11     .00000         13.025   11  1937     .00000
 (3) vs (1)   134.965      4     .00000         34.847    4  1945     .00000
 (4) vs (1)   298.013     15     .00000         21.290   15  1934     .00000
 (4) vs (2)   158.932      4     .00000         41.057    4  1934     .00000
 (4) vs (3)   163.047     11     .00000         15.333   11  1934     .00000


Random Effects Model: v(i,t) = e(i,t) + u(i)
2 estimates of Var[u] + Q * Var[e]
Based on     Means             OLS
        .12341D-01      .18472D-01
(Used  Means.  Q  =     .0063)
Estimates:  Var[e]              =   .122751D+00
            Var[u]              =   .115681D-01
            Corr[v(i,t),v(i,s)] =   .086124
Lagrange Multiplier Test vs. Model (3) =     754.43021
( 1 df, prob value =  .000000)
Fixed vs. Random Effects (Hausman)     =        .00010
( 4 df, prob value = 1.000000)
Estd. Autocorrelation of e(i,t)    -.014355
Reestimated using GLS coefficients:
Estimates:  Var[e]              =   .122754D+00
            Var[u]              =   .186070D-01
            Sum of Squares          .258753D+03
            R-squared               .645264D-01
Variable  Coefficient  Std. Error   t-ratio Prob|t|Úx   Mean of X  Std.Dev.of X
---------------------------------------------------------------------------
PRIMERINTEN.15949        .2473E-01    6.449   .00000   .29979        .46851
ORDTAREA  -.15046E-01    .5473E-02   -2.749   .00598   3.0117        1.4662
INTENTO   19-.18479E-01    .9952E-02   -1.857   .06334   2.4228        1.2244
ACCION     .13265E-01    .2689E-02    4.932   .00000   9.1824        2.9569
Constant   .37312        .5234E-01    7.128   .00000
```

Tabla 6.23: Resultados para la variable tiempo con errores.

El modelo más adecuado es el aleatorio (Hausman=.00010; p=1). En este caso, el factor de interés aparece como significativo (t=4.932; p<0.00001) con un coeficiente de 0.0132. El gráfico que muestra los efectos para los valores considerados en la escala normal aparece a continuación. Cada nueva acción aumenta el tiempo necesario para realizar la tarea en aproximadamente 0.09s.

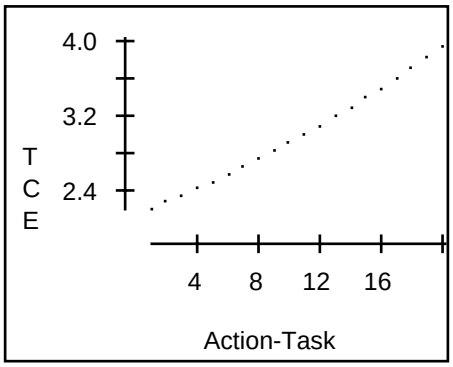


Figura 6.29: Efecto de la variable Action-Task

Los gráficos de residuales se muestran a continuación:

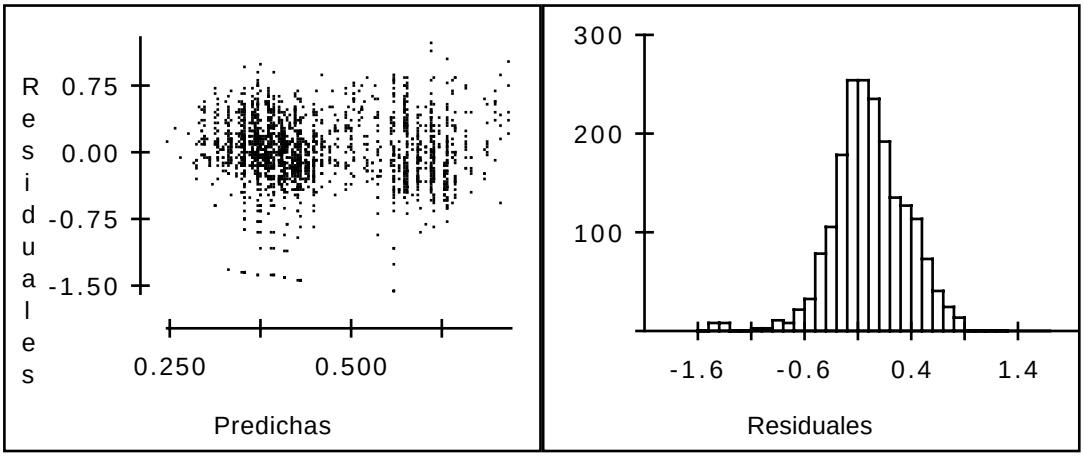
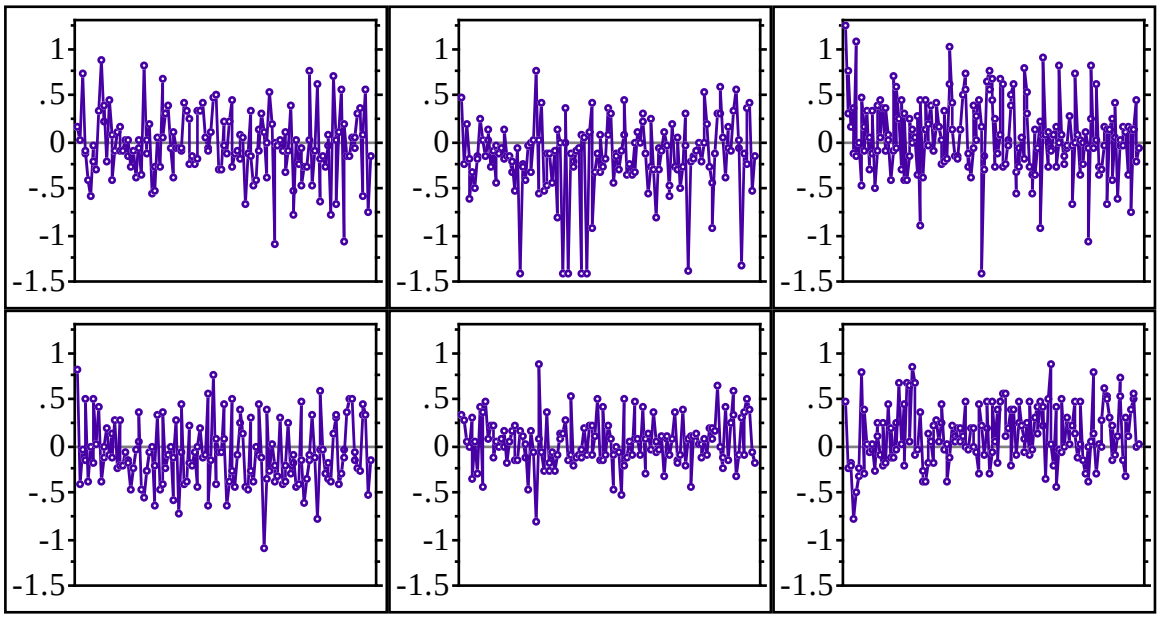


Figura 6.30: Graficos de residuales para la variable dependiente con error.

Se aprecia una ligera heteroscedasticidad central, y los residuales son bastante simétricos (asimetría=-0.53) aunque existen varios valores extremos negativos.



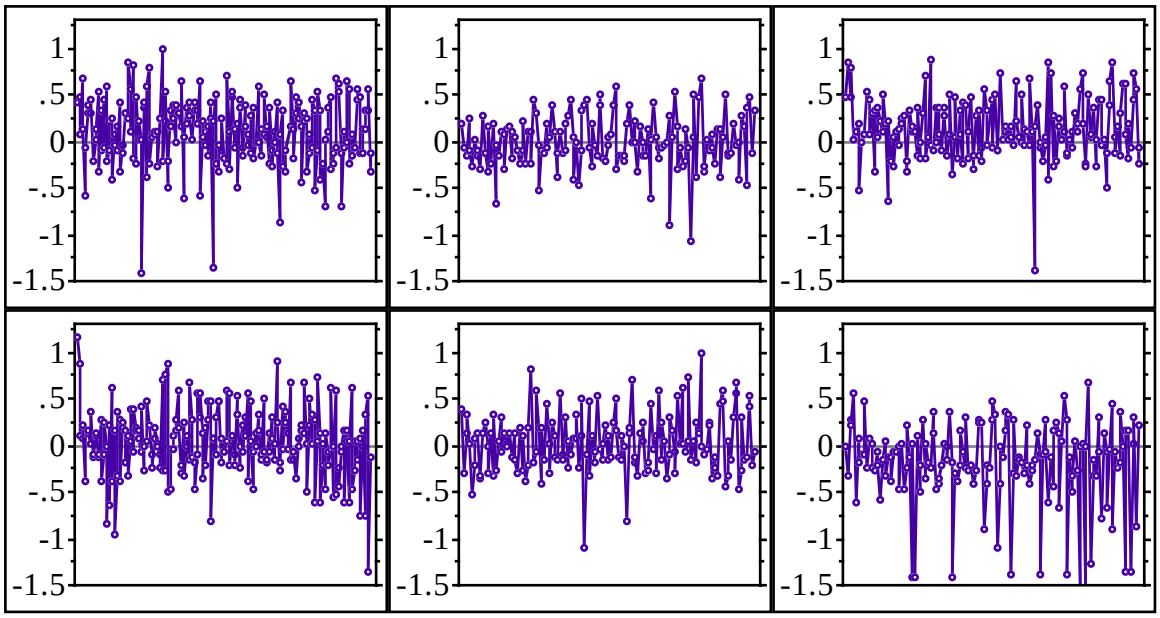


Figura 6.31: Gráficos de residuales por series temporales para la variable tiempo con error.

Resultados no mostrados aquí revelaron como la transformación realizada eliminó la mayor parte de la autocorrelación intrasujeto para todos los individuos hasta el nivel 14.

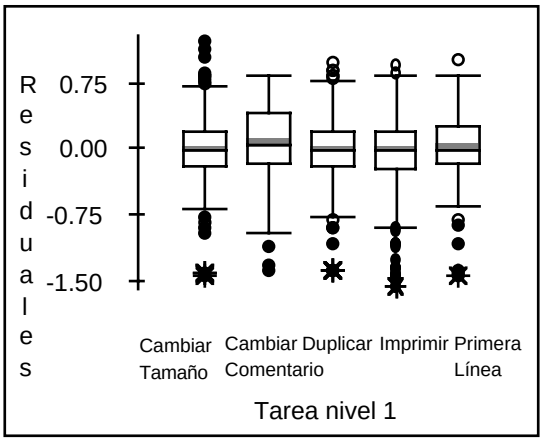


Figura 6.32: Gráfico de cajas para la tarea 1.

Fuente	gl	SC	MC	F	Prob
Const	1	0.099020	0.099020	0.74644	0.3877
Tn1	4	2.67466	0.668664	5.0406	0.0005
Error	1957	259.608	0.132656		
Total	1961	262.283			

Tabla 6.24: Análisis de varianza para los residuales.

Los resultados muestran diferencias entre las tareas con respecto a la explicación de los residuales. Las pruebas a posteriori muestran que, de nuevo, Comentario parece tener mayores residuales positivos que otras tareas.

	Diferencia	err. típ.	Prob.
Guardar Como - Comentario	-0.083177	0.0283	0.033117
Imprimir - Comentario	-0.119386	0.0276	0.000161
Imprimir - Guardar Como	-0.036209	0.0244	0.771356
Prim. linea - Comentario	-0.054925	0.0294	0.472573
Prim. linea - Guardar Como	0.028252	0.0264	0.964697
Prim. linea - Imprimir	0.064461	0.0256	0.112673
Tipo tamaño - Comentario	-0.082453	0.0281	0.032884
Tipo tamaño - Guardar Como	0.000724	0.0249	1.00000
Tipo tamaño - Imprimir	0.036933	0.0240	0.735727
Tipo tamaño - Prim. linea	-0.027528	0.0261	0.968102

Tabla 6.25: Pruebas a posteriori para los residuales.

Capítulo 7

Conclusiones

En este capítulo se llevará a cabo una descripción de las conclusiones alcanzadas en este trabajo. Cada objetivo será discutido individualmente, siendo los resultados de su discusión el punto de arranque del siguiente objetivo. Un apartado de discusión general cerrará finalmente el capítulo.

7.1. Objetivo 1:

Evaluar qué metodologías permiten tanto la representación del funcionamiento de un interfaz como el análisis de sus características.

El primero de los objetivos de este trabajo explora qué tipo de notaciones podrían ser útiles para realizar descripciones de sistemas que a su vez fueran susceptibles de ser analizadas mediante procedimientos establecidos. En el marco de este objetivo nos proponemos seleccionar aquellas técnicas que son capaces de satisfacer lo mejor posible la necesidad que diseñador tiene de plantear el interfaz que desea construir y luego depurarlo y mejorarlo.

Algunas de las técnicas existentes en la literatura se encuentran en un polo de los extremos siguientes. O bien proporcionan fundamentalmente un mecanismo de representación que luego culmine en la implementación, o, por el contrario, toman artefactos ya casi completamente especificados para

realizar análisis que los mejore. Nuestro objetivo era determinar qué técnicas podrían ofrecer un marco en el que se diera cabida a ambos extremos.

En la figura 7.1 es posible ver una representación que puede servir de punto de partida para la discusión acerca de los modelos que consideramos que tienen las características apropiadas. Como es posible ver, dos dimensiones han sido utilizadas para realizar una clasificación que distingue entre: 1) modelos centrados en el usuario frente a modelos centrados en el sistema y 2) profundidad v. superficialidad en la forma que estos modelos son analizados.

Esto nos lleva a los siguientes cuatro tipos:

• Modelos que analizan el sistema con profundidad: Estos modelos intentan realizar deducciones lógicas acerca del espacio de posibilidades que determinados planteamientos pueden configurar. Estos son los modelos del sistema descritos en el capítulo 2 y 3 y permiten realizar análisis acerca de, por ejemplo, qué consecuencias y requisitos tendrá la acción de deshacer en un programa de trabajo en grupo, o cómo podría lograrse que los resultados de todas las acciones del usuario fueran predecibles. El modelo PiE es probablemente el más extremo en esta categoría, ya que trabaja antes con las propiedades de los sistemas que con sistemas concretos. El modelo de agente en cambio sería un ejemplo de notación más cercana a la descripción de sistemas concretos.

• Modelos que describen sistemas concretos sin entrar en las profundidades del espacio de problemas de clases de sistemas. Los sistemas de descripción de diálogos permiten establecer el funcionamiento de sistemas concretos, pero muy a menudo esto significa una cantidad de abstracción inferior a la que es posible obtener mediante los modelos del grupo anterior. Ello significa ser más superficial en el análisis, pero más explícito en la descripción, estando, por tanto, más cercanos a la implementación que los modelos anteriores. Su valor fundamental estriba en permitir representar las consecuencias de las acciones de los usuarios sobre el sistema y las acciones que serán permitidas a éstos.

• Modelos superficialmente cognitivos. Estos modelos están muy relacionados con los anteriores pero su diferencia fundamental radica en dar un pequeño paso dentro del usuario y empezar a plantearse qué ocurre dentro de éste entre cada acción aceptable para el sistema. Esto corresponde a

modelos tipo GOMS, y son denominados superficialmente cognitivos porque, cuando sería necesario hacer referencia a un proceso mental de índole complicada, se sustituye por una llamada genérica que no intenta profundizar en la problemática subyacente. TAG es un modelo que se sitúa también en ese rango en nuestra opinión aunque incorpora una serie de suposiciones acerca de la descripción de la consistencia entre tareas.

• Modelos profundamente cognitivos: Estos modelos dan un paso adelante y se interesan sobre la manera en que los sujetos comprenderán el funcionamiento del interfaz, los conceptos que manejarán en relación a él, así como los problemas que ello les puede originar. PUM por ejemplo se planteará el "programa del usuario" el cual desembocará en una serie de acciones que pueden o no ser correctas desde el punto de vista del interfaz. Este tipo de análisis es relativamente complejo y difícil de plantear, y es muy posible que buena parte de los análisis realizados se apoyen en una clase de conocimiento propio de los investigadores que desarrollaron la técnica, y, por tanto, poco transferible para su uso general.

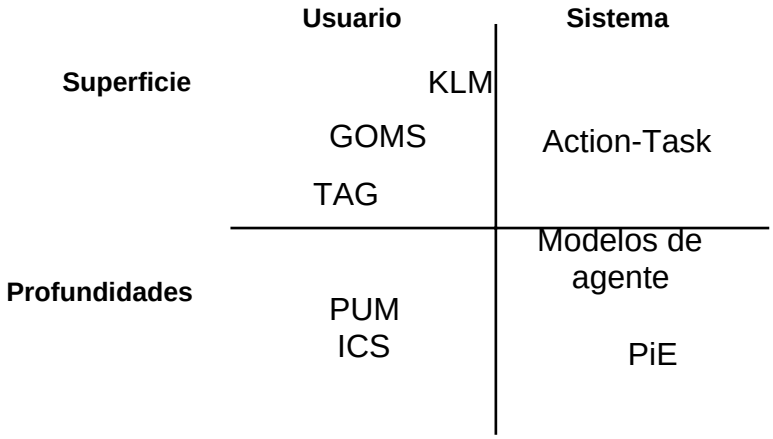


Figura 7.1: Una clasificación de los modelos formales

A partir de esta clasificación, nuestra respuesta a la pregunta efectuada acerca de qué tipo de notaciones podrían ser utilizadas para realizar tanto descripciones como análisis de interfaces en situaciones de diseño concretas, es que, en la actualidad, sólo las notaciones situadas a un nivel superficial en cuanto al sistema o al diálogo parecen apropiadas. Esto se debe a que tanto los modelos del sistema o los modelos cognitivos que pretenden análisis profundos carecen de características adecuadas para satisfacer los requerimientos de representación. Veamos por qué:

• Los modelos del sistema analizan problemas desde un punto de vista general, lo cual hace que las cuestiones de implementación concreta que un diseñador debería especificar no son tratados, quedando la descripción como una forma de captar la dinámica general de funcionamiento. Los modelos formales a este nivel parecen herramientas para generar teorías o normas generales antes que para solucionar problemas concretos de diseño, o quizás para convertir el diseño en algo más guiado por principios, pero no para especificar diseños concretos.

• Los modelos cognitivos del usuario más profundos adolecen en general de una falta de capacidad de representación del sistema. Dado que su objetivo es fundamentalmente el análisis de los prototipos de diseños, no ofrecen ningún método para representar éstos, por lo que su utilidad en un contexto de evaluación formal requiere la existencia de métodos independientes que permitan especificar de algún modo el comportamiento del sistema.

TAG y NGOMS, notaciones que nosotros hemos considerado como superficialmente cognitivas, permiten en cambio cierto grado de representación del sistema así como de análisis. Sin embargo, la parte de representación no está tan desarrollada como en las técnicas de representación de diálogos. No obstante, aunque de una manera moderada, utilizando estas notaciones es posible obtener una idea acerca de cómo es las la dinámica del sistema propuesta por el diseñador. Aún así, todavía quedaría un grado de ambiguedad quizás demasiado grande.

Por otro lado, con respecto a las notaciones para la descripción de diálogos, estos métodos tienen tanto un componente de especificación como de análisis. Ahora bien, en este caso la parte de análisis es la menos desarrollada, siendo un tipo de notaciones más apropiadas para la representación. Por otra parte, han sido sugeridas en ocasiones una serie de recomendaciones que permitirían una serie de análisis sobre las representaciones realizadas. De este modo, sobre la misma especificación podrían extraerse conclusiones que pudieran ser utilizadas para mejorar el diseño. Ahora bien, el número de sistemas propuestos en la literatura para la representación de diálogos es muy grande por lo que finalmente se decidió explorar sólo uno de ellos.

En definitiva, las siguientes notaciones fueron analizadas en el siguiente paso: NGOMS, TAG, Action-Task, PUM y ICS. Estas dos últimas, como hemos visto, no deberían haber sido revisadas debido a que sus

características no ajustaban al perfil de técnicas que deseábamos estudiar. No obstante, otras cuestiones tales como su novedad y su impacto en la literatura nos hicieron interesarnos por ellas en la segunda parte de este trabajo.

7.2. Objetivo 2

Evaluación de la capacidad de las distintas notaciones para ser utilizadas efectivamente en la descripción de prototipos de diseños.

Con los modelos seleccionados se intentó realizar la descripción de un interfaz, de modo que se pudieran evaluar la facilidad/dificultad de estas técnicas. Los resultados serán comentados para todas las técnicas atendiendo a los siguientes puntos:

• Documentación: Para poder utilizar un modelo con comodidad es necesario que haya un mínimo de documentación que cubra los distintos aspectos en relación con la utilización de la técnica, aclare y resuelva los puntos conflictivos, proponga ejemplos y los discuta en profundidad, etc. En la mayoría de los casos, las técnicas comentadas han sido principalmente expuestas en artículos científicos, dirigidos a ahondar sobre cuestiones relacionadas con la técnica antes que en proporcionar medios sencillos para comprender y utilizar correctamente sus posibilidades. Además, en muchos casos, los ejemplos manejados no corresponden a aplicaciones de tamaño real, sino a simplificaciones construidas sólo para ilustrar la aplicación de las herramientas manejadas.

• Facilidad de uso: Hay bastantes cuestiones que ayudan a la facilidad de uso de las diversas técnicas:

-El tamaño de las descripciones resultantes: Ciertas técnicas exigirán un nivel de detalle que es inabordable en situaciones prácticas.

-Existencia de herramientas informáticas o de otro tipo que ayuden a la especificación.

• Claridad de conceptos: Ciertas técnicas dejarán puntos ambiguos que el diseñador tendrá que decidir en cada ocasión. Estas decisiones son por un lado arbitrarias pero tan importantes que pueden determinar por completo el análisis realizado. El ejemplo más claro es la unidad de análisis o grado de agregación de las tareas simples.

• Análisis: Los análisis que es posible realizar por medio de las técnicas pueden ser de varios tipos. Análisis cualitativos que muestren fallos en el diseño planteado. Análisis cuantitativos que indiquen variables que podrían ser modificadas para mejorar el diseño. Además, ciertas técnicas ofrecerán la posibilidad de realizar análisis a diferentes niveles, así ofreciendo la posibilidad de obtener información acerca de muchas cuestiones de posible interés.

• Problemas de representación: Ciertas técnicas no serán capaces de representar ciertas situaciones.

• Nivel de actividad en esta linea de trabajo: Algunas de las técnicas consideradas aquí han sido desechadas por los autores que las propusieron por lo que resulta seguramente poco interesante seguir intentando utilizarlas a menos que uno se sienta satisfecho con lo que tiene o bien desee realizar nuevas propuestas.

A continuación comentaremos estos puntos en relación con las técnicas utilizadas:

1) NGOMS

• Documentación: Este método es el que posee una documentación probablemente más completa de todos los considerados. Fundamentalmente el tutorial de Kieras y Elkerton (Kieras, 1991) proporciona descripciones muy accesibles, con ejemplos detallados y sugerencias para el análisis cualitativo. Otra información acerca de CCT o GOMS (predecesores de NGOMS) no es en cambio tan clara. Muchos autores han comentado que la notación propuesta en el libro de Card, Moran y Newell (Card, et al., 1983) es bastante ambigua y poco fácil de utilizar. En cuanto a CCT, existe un artículo (Bovair, et al., 1990) que explica los fundamentos y conceptos teóricos detrás de esta técnica pero no existen desarrollos bien documentados aparte de los ejemplos allí manejados.

• Facilidad de uso: NGOMS es una metodología fácil de usar. Las descripciones son simples y las convenciones son comprensibles rápidamente. El tamaño de las descripciones resultantes no es excesivamente grande y no resulta costoso realizarlas. En la actualidad se están empezando a desarrollar herramientas informáticas (v.p.e (Byrne, et al., 1994)) aunque todavía no están lo suficientemente avanzadas como para ser utilizadas en práctica

• Claridad de conceptos: Los conceptos en NGOMS son sencillos comparados con otras metodologías.

• Análisis: NGOMS tiene la mayor amplitud en cuanto a resultados del análisis en principio. Por medio de análisis basados en esta metodología es posible obtener predicciones cuantitativas y cualitativas de la usabilidad del interfaz. Además, varias variables dependientes pueden ser consideradas: tiempo de aprendizaje, tiempo de ejecución y carga mental.

En nuestro trabajo, no obstante, las predicciones cualitativas fueron poco numerosas y además no se confirmó su importancia en los análisis realizados. Ello no significa sin embargo que gracias al ejercicio realizado no se captaran elementos de la dinámica del interfaz que no resultaran de interés. Sin embargo, estos elementos no fueron considerados como directamente extraídos del ejercicio realizado, ya que aparentemente cualquier otros esfuerzo de análisis que hubiera conducido a una evaluación minuciosa habría probablemente producido el mismo resultado.

• Problemas de representación: En nuestro trabajo hemos encontrado que el mayor problema de NGOMS en cuanto a representación se encuentra en que su estructura está dirigida por objetivos, por lo que todas aquellas acciones no directamente conectadas con estos objetivos no son representadas. Esta falta de información es un problema de representación que la hace poco adecuada para llevar a cabo implementaciones en situaciones reales.

NGOMS no permite descripciones a niveles pequeños de detalle. Es decir, no posee un equivalente a KLM (Keystroke Level Model) tal y como el original GOMS tenía, que le permita asumir descripciones en las que la existencia de una estructura basada en objetivos resulte cuestionable. Esta necesidad de un segundo modelo proviene del hecho que, a medida que las descripciones están más cerca del ser humano, los procedimientos a describir están más influidos por la existencia de objetivos, mientras que, a medida que la descripción tiene un nivel inferior, nos aproximamos a las peculiaridades y condicionantes de la máquina y la tarea, los cuales no se ajustan a una descripción de este tipo. Esta dicotomía estaba presente en los primeros trabajos acerca CCT (predecesor de NGOMS), en los que se introducía una segunda notación para tratar con estos aspectos. En las últimas versiones de NGOMS no hay un equivalente semejante y, cuando la descripción

empieza a alcanzar niveles cercanos a los detalles de implementación el diseñador debe detenerse.

• Línea de trabajo activa/discontinuada: NGOMS es una de las líneas de trabajo en relación con el análisis por medio de métodos formales más activa que existe en la actualidad.

2) TAG

• Documentación: La documentación en TAG existe solamente en forma de publicaciones de carácter científico que explican la notación. En uno de ellos no obstante (Schiele y Green, 1990) se hace una explicación bastante exhaustiva de la técnica y es por tanto la mejor fuente de información disponible acerca de la misma. El ejemplo manejado en ese artículo es de tamaño real (no un ejemplo de tamaño reducido) e incluye tres programas completos pudiendo ser quizás uno de los ejemplos más ambiciosos manejados en la literatura.

• Facilidad de uso: TAG no es difícil de manejar. Las descripciones resultantes tienen un tamaño bastante reducido aunque no hay hasta la fecha ninguna herramienta informática que ayude a la especificación de modelos mediante esta notación.

• Claridad de conceptos: En TAG existe un elemento de ambiguedad que proporciona un grado de inseguridad bastante amplio a las descripciones realizadas. Debido a que el diseñador necesita postular cómo los usuarios percibirán la consistencia entre acciones a partir de sus propias intuiciones, los análisis tienen un componente que vicia las conclusiones que puedan alcanzarse. Es muy posible que esa particular manera en que el diseñador espera que los usuarios perciban el interfaz no sea la que los usuarios capten y por tanto no se produzca el efecto de consistencia esperado.

• Análisis: TAG ofrece básicamente información sobre si la tarea analizada es consistente con otras o no lo es. Esta información es básicamente cualitativa y puede servir para detectar tareas que puede esperarse que resulten problemáticas para el usuario. Esta es una información limitada pero interesante. De nuevo, realizar la descripción puede resultar en una serie de intuiciones sobre el posible comportamiento de los usuarios con el sistema, pero no existe información explícita acerca de este extremo.

Una cuestión no tratada son los aspectos de evaluación por parte del usuario de los efectos de sus acciones. Las modificaciones a TAG aparecidas posteriormente han fortalecido este punto.

• Problemas de representación: TAG es en nuestra opinión demasiado estático con respecto tanto al sistema descrito como al usuario. Así, por un lado carece de una manera de representar la dinámica del sistema, y, por otro, no muestra la lógica de las acciones de los sujetos. Esto se debe a que esta notación se dirige solamente a captar un concepto, la consistencia, y no considera la representación de otro tipo de cuestiones. Otro gran problema es que no entra en la representación de la información que el sujeto recibe a la hora de realizar sus acciones y que le va formando en la generación de sus conceptos, los cuales pueden ser los determinantes de la percepción de la consistencia.

• Línea de trabajo activa/discontinuada: TAG es una linea de acción discontinuada. En la actualidad, varias técnicas y notaciones que han surgido en relación o a partir de esta notación se han convertido en el centro de interés de los autores que propusieron estas técnicas.

3) *Action-Task.*

• Documentación: La documentación acerca de Action-Task consiste solamente en artículos que introducen el método, no existiendo información detallada acerca de ejemplos u otro tipo de ayudas.

• Facilidad de uso: La notación es fácil de utilizar, pero a medida que se desea realizar una descripción a un nivel detallado, éstas no son demasiado fáciles de seguir cuando la descripción ha sido realizada textualmente. No obstante, realizar las descripciones no es excesivamente complicado.

Hay que tener en cuenta por otro lado la existencia de una herramienta informática por medio de la cual es posible realizar las descripciones. Esta herramienta permite mucho más fácilmente seguir la dinámica de funcionamiento del interfaz, pero, a cambio, es mucho más problemática para realizar descripciones. Esto se debe a que, debido a que esta herramienta funciona en una hoja de cálculo plantea ciertas limitaciones prácticas tales como el tamaño de la pantalla en la que se desea visualizar el resultado. Por otro lado, es posible obtener ciertas medidas de complejidad del interfaz de un modo automático o semiautomático, tal como el número de acciones disponibles tras cada acción.

No obstante, hay que señalar que la forma en que ha sido utilizada esta notación difiere de la sugerida por los autores que la proponen, ya que estos recomiendan sólo representar el nivel superior de la dinámica del sistema en lugar de los detalles más concretos. En nuestro caso, al representar los detalles más concretos, los problemas encontrados pueden ser diferentes a los que los autores habrían sugerido.

• Claridad de conceptos: Esta notación es relativamente fácil de entender y utilizar.

• Análisis: La parte de análisis no está bien desarrollada ya que no es el objetivo central de esta notación. Los autores sugieren que una vez realizada la descripción se compruebe ésta con una serie de sugerencias, fundamentalmente las referidas a la completud de la descripción, a la no existencia de tareas difíciles de realizar, contraste con normas de buen diseño, etc. Estos análisis no obstante no han sido desarrollados en extensión sino que permanecen fundamentalmente dependientes de las intuiciones de los diseñadores.

• Problemas de representación: La notación presenta varios problemas de representación. Uno de ellos es general a todas las notaciones basadas en reglas de producción y hace referencia a los problemas que tiene a la hora de representar sistemas que fundamentalmente tienen un comportamiento secuencial. Notaciones basadas en diagramas de transición o de otro tipo son más convenientes para esta tarea.

Un segundo problema encontrado es la representación de entidades que podrían ser descritas a un grado inferior o dejadas sin describir en detalle. Por ejemplo, si queremos describir el interfaz de una aplicación, ésta podría tener un elemento de interfaz denominado barra de menús (que tendría un comportamiento guiado por una serie de reglas), a partir del cual se podría aludir a un cuadro de diálogo en el cual aparecerían botones u otros elementos. Estas entidades podrían ser descritas de un modo independiente y se podrían hacer llamadas en caso de necesitar un elemento inferior (por ejemplo, el cuadro de diálogo podría incluir llamadas a botones sin necesidad de especificar los detalles acerca de mover el ratón dentro de ellos, quitar el ratón sin haber soltado el botón, etc.). Este tipo de problemas son difíciles de solucionar con el sistema actual aunque es posible ver con relativa facilidad cómo podrían ser abordados. Esto proporcionaría la oportunidad de tener módulos reutilizables en descripciones posteriores (a modo de bibliotecas de rutinas).

Un tercer problema de representación es el que ocurre cuando la condición que dispara una acción tiene más de dos valores posibles (verdadero y falso). Por ejemplo, un sistema de comprobación de ortografía podría tener tres diccionarios (p.e. inglés, francés, español). En el sistema actualmente utilizado sería necesario indicar tres reglas, una por cada diccionario, que indicarían qué diccionario está conectado en cada momento y qué acciones serían posibles como consecuencia de ello. En un sistema que aceptara

condiciones más complejas, se necesitaría una sola regla (con precondiciones posibles a inglés, francés y español).

• Línea de trabajo activa/discontinuada: Esta línea de trabajo no ha sido discontinuada, habiéndose producido una serie de mejoras sobre la línea inicial de manera continua.

4) PUM

• Documentación: La documentación en relación con PUM sólo existe en forma de artículos que muestran ejemplos concretos sobre las posibilidades de este método de descripción. Estos ejemplos no tienen como objetivo describir en detalle las características de la notación, sino que se dirigen a profundizar en las capacidades que ésta posee.

• Facilidad de uso: Según los autores que propusieron esta técnica, sus características son lo suficientemente complejas como para que sea difícil pensar que los propios diseñadores encargados de construir prototipos sean capaces de realizar estos análisis por sí mismos. Así, es necesario pensar en la existencia de algo llamado un "PUMer" o persona especializada en realizar este tipo de análisis (Shum, 1994).

En relación con las herramientas informáticas, como ya ha sido comentado anteriormente, éstas no pudieron ser utilizadas para realizar ninguna descripción debido a su carácter todavía experimental y a la inherente complejidad que presentan ya que se componen de: 1) un lenguaje de descripción propio para el que no existe en la actualidad ninguna documentación, 2) un programa que funciona en LISP y que traduciría ese lenguaje al lenguaje de SOAR (una herramienta de inteligencia artificial) y 3) el interpretador de SOAR que permitiría analizar el resultado. Por ello, a pesar del interés que esta herramienta posee no ha sido posible obtener ningún resultado satisfactorio.

• Claridad de conceptos: Los conceptos en PUM no son obvios. Al parecer, un amplio conocimiento en Psicología Cognitiva, SOAR y experiencia en Interacción Hombre-Ordenador son condiciones importantes para intentar utilizar esta técnica.

• Análisis: Esta técnica es fundamentalmente analítica. Sus resultados deberían producir intuiciones cualitativas acerca de posibles problemas que los usuarios podrían tener con el interfaz.

• Problemas de representación: PUM no permite la representación del dispositivo. Su objetivo es representar a un usuario que utiliza el dispositivo y por tanto este tiene que ser descrito previamente.

• Línea de trabajo activa/discontinuada: Esta línea de trabajo se ha caracterizado por una gran cantidad de trabajo en relación al desarrollo del proyecto ESPRIT Amodeus. La finalización de este proyecto en el año 1995 deja abierta la cuestión acerca de su continuidad.

5) ICS

• Documentación: La documentación acerca de ICS existe en artículos que explican el funcionamiento de esta metodología, pero existen varios problemas añadidos en este caso a este hecho.

En primer lugar, dado que se trata de un sistema experto y por tanto reside en un computador resulta completamente opaco a la observación, por lo que no hay una manera sencilla de entender los principios en los que se basa. Estos principios se encuentran en los artículos que explican la teoría detrás de ICS pero debido a la complejidad de la propia teoría o quizás a su excesiva amplitud resulta difícil saber hasta que punto el sistema experto se relaciona con esta teoría. Por otro lado, mientras que PUM se apoya en una arquitectura cognitiva desarrollada independientemente y con una gran cantidad de investigación detrás de ella, ICS es, además de un sistema de análisis de la interacción hombre ordenador, una arquitectura cognitiva propia que, muy probablemente, necesita todavía un largo camino antes de convertirse en una propuesta suficientemente estable.

En segundo lugar, el sistema experto no tiene tampoco documentación que explique la forma de manejarlo, dado que se trata fundamentalmente de una "demostración del concepto", y no de un producto acabado.

• Facilidad de uso: Las descripciones realizadas por medio de ICS son en principio las que resultarían más fáciles de realizar ya que al tratarse de un sistema automatizado basado en menús, los usuarios deberían de poder realizar fácilmente las descripciones simplemente siguiendo las instrucciones hasta completar éstas. No obstante, el sistema informático posee muchos problemas de usabilidad propios, tales como preguntas ambiguas, confusión entre niveles o pasos y otros puramente informáticos (fallos de funcionamiento, etc.). Ello lo convierte en relativamente difícil de usar a pesar de las aparentes ventajas que ofrece.

• Claridad de conceptos: Como ya ha sido comentado, no resulta fácil entender muchas de las preguntas que hace el sistema. En muchos casos las respuestas no parecen muy apropiadas a la situación y en otras simplemente parece que se esté contestando a la misma pregunta continuamente.

• Análisis: El área principal de ICS es el análisis. Al respecto no existen resultados positivos. Los autores originales afirman que la utilización de esta técnica les reporta muchas ventajas. Por el contrario, información proporcionada en estudios de utilización del sistema experto por diseñadores, éstos fueron poco positivos acerca del sistema, opinando que la información obtenida era evidente por sí misma y el uso del sistema experto era poco menos que una actividad irrelevante para su obtención. (Shum y Hammond, 1993). En nuestro caso resulta difícil realizar un diagnóstico de las posibilidades de análisis del sistema experto debido al reducido número de resultados obtenidos.

• Problemas de representación: ICS no es una estructura diseñada para representar interfaces sino que está dirigida principalmente al análisis.

• Línea de trabajo activa/discontinuada: Del mismo modo que la línea de trabajo anterior, durante los dos últimos años ha habido una gran actividad en relación a esta técnica en coincidencia con un proyecto ESPRIT que finalizó este mismo año.

7.3. Objetivo 3

Evaluación de las predicciones obtenidas a través de las notaciones finalmente seleccionadas en un ejemplo de diseño.

En este apartado se ofrecerá una discusión acerca de los resultados obtenidos a la hora de realizar predicciones del tiempo empleado por los sujetos en resolver las tareas propuestas y una discusión acerca del valor de los resultados encontrados.

En relación con ello, se discutirá en primer lugar la problemática de las comparaciones entre las distintas metodologías. En segundo, el interés de la estructura de análisis estadísticos como marco para la realización de estudios similares y las ventajas sobre otros estudios que realizan aproximaciones diferentes. Asimismo, comentaremos los resultados obtenidos con respecto a la variable dependiente utilizada. Por último, comentaremos los resultados obtenidos para cada uno de los niveles de análisis considerados y el valor que estos pueden tener en situaciones aplicadas de diseño.

1) El problema de los niveles de análisis a la hora de realizar comparaciones.

Uno de los posibles resultados deseados en relación a este trabajo consistía en realizar una comparación entre las diversas técnicas de análisis formal con objeto de intentar determinar si alguna de ellas era capaz de realizar predicciones más ajustadas, y, por tanto, podía convertirse en la técnica a preferir sobre las otras.

No obstante, estas comparaciones progresivamente se manifestaron llenas de complejidad debido a los diferentes granos de análisis sobre los que estas metodologías eran capaces de realizar predicciones. Cada técnica era capaz de realizar predicciones sobre niveles distintos de ejecución de los usuarios y sólo en ciertas situaciones fue posible hacer coincidir las predicciones de dos metodologías distintas sobre un mismo problema, logrando de este modo una comparación efectiva entre ellas. No obstante, incluso en este caso, nuestra experiencia sugiere que cada técnica apunta a cuestiones diferentes, y que, por tanto, ambas pueden ser interesantes a la hora de realizar predicciones y mejoras en el interfaz.

NGOMS, por ejemplo, fue capaz de trabajar a dos niveles en nuestra opinión. Un nivel global, en el que las tareas predichas fueron definidas al mismo nivel que les fueron explicadas a los sujetos, y uno inferior, en el que se definieron subtareas a un nivel de agregación inferior.

TAG en cambio no pareció apropiado para tareas que estuvieran a un nivel superior, ya que, partiendo de acciones situadas a un nivel inferior, las colapsa a un nivel superior, no teniendo sentido ascender hasta llegar a un nivel de colapsamiento superior. Ello hace que las predicciones realizadas con este método coincidan con NGOMS en ese nivel intermedio.

Action-Task es el sistema de más bajo nivel de todos los utilizados. Su rango de acción incluye acciones concretas a nivel del interfaz tal y como movimientos de ratón, etc. El resultado es que los análisis realizados lo son del propio entorno de la tarea, antes que los de un usuario intentando realizar esa tarea e incorporando un número de limitaciones y características cognitivas a ese trabajo. En lugar de decidir que una característica es importante desde el punto de vista del usuario y construir una notación completa alrededor de ella, la idea es construir una representación del dispositivo. Esta representación más tarde podría ser analizada desde diferentes puntos de vista o características que fueran consideradas

importantes. En nuestro caso, el número de acciones posibles ha sido la característica utilizada, pero es posible pensar en otras características que podrían ser examinadas en el modelo.

Podría plantearse la agregación de datos como una solución a este problema. Esta agregación debería realizarse en los modelos de menor grano hacia los modelos de mayor grano. No obstante, esta agregación es probablemente poco razonable en la mayoría de los casos. Por ejemplo, supongamos que TAG ofrece una tarea como inconsistente (nivel intermedio) y realizamos una agregación a un nivel superior de tal modo que tareas consideradas como consistentes y otras como inconsistentes son agrupadas en una misma unidad a ese nivel. ¿Cómo deberíamos en este caso tratar la inconsistencia en una subtarea? ¿Deberíamos calcular algo así como una proporción o porcentaje, en este caso del número de tareas consistentes o inconsistentes (a nivel intermedio) de que está compuesta esa tarea a un nivel superior? La respuesta no es obvia y, en nuestra opinión, es un ejemplo de la falta de sentido que supone realizar este tipo de agregaciones.

Otro argumento más en contra de la agregación de datos sería que un valor agregado posiblemente no es representativo de la dinámica interna que lo ha producido. Por ejemplo, una acción del interfaz puede ser manifiestamente problemática y otra extraordinariamente cómoda. Agregar los tiempos escondería este hecho e impediría captar la posiblemente diferente dinámica que podría subyacer.

2) La estructura de análisis estadísticos puede servir como un marco metodológico para estudios similares

Al llevar a cabo un estudio sobre la usabilidad de un nuevo producto resulta natural plantearlo por medio de la asignación de una serie de tareas a un grupo de sujetos, los cuales aprenderían a manejar el interfaz mientras el diseñador registra su comportamiento. A partir de este registro, es posible extraer información cualitativa que puede ser de indudable interés. Sin embargo, en ocasiones el diseñador puede querer un análisis cuantitativo que muestre qué desviación se produce en ciertas tareas respecto a los resultados esperables. Esta información, no obstante, está afectada por una serie de variables (aprendizaje, diferente capacidad en sujetos, primeros intentos, etc.) que dificultan la obtención de resultados interpretables y que el investigador, movido por interés aplicados o teóricos, necesitaría controlar.

El análisis de las desviaciones respecto a valores esperables proporciona un mecanismo para evaluar prototipos únicos, frente al método de comparaciones experimentales en el que dos o más prototipos son comparados para de este modo decidir cuál de ellos es mejor. Dado que en la mayoría de los casos sólo existe un prototipo, este tipo de análisis podría ser especialmente interesante. Utilizar técnicas desarrolladas dentro del enfoque del análisis exploratorio de datos (incluir variables ficticias para situaciones claramente conceptualizadas como diferentes por ejemplo) podrían complementar estos análisis.

En este método de análisis, los sujetos realizarían una serie de tareas seleccionadas sobre el prototipo desarrollado. Dado que inevitablemente ciertas tareas serán probadas antes que otras, y dado que ello conllevará posiblemente un efecto de autocorrelación que es necesario controlar, el conjunto de variables utilizadas en nuestros análisis pueden ser de gran relevancia (primer intento, orden de la tarea, etc.). Asimismo, en caso de tener un método formal que realiza predicciones empíricas, éstas pueden ser incluidas como un mecanismo adicional de explicación de los resultados. *Aquellas tareas o momentos que destaquen por sus valores excesivamente altos con respecto al análisis de series temporales ponderadas, incluyendo todas esas variables, son buenas candidatas para un rediseño.* Variar el orden en el que las tareas son ejecutadas puede ser un recurso insuficiente, ya que en muchas ocasiones este orden vendrá dado por la propia naturaleza de las tareas, y no será posible variarlo entre los sujetos.

Obviamente, este tipo de análisis parte de la idea de tener un número de tareas comparables entre sí y con efectos sobre la variable dependiente equivalentes. En caso de no ser así, las fuentes de variabilidad de estas tareas deberían ser controladas, bien mediante variables correlacionadas con ellas (básicamente el enfoque utilizado en este estudio), o bien mediante variables ficticias. Por ejemplo, una tarea podría necesitar una cantidad de tiempo relativamente grande debido a que el mecanismo de input es engorroso, pero no existe ninguna manera de evitar ese input. Este valor podría incluirse en un análisis NGOMS, o simplemente ser restado al realizar la recogida de datos. No obstante, en ocasiones puede que ese tiempo sea de difícil cálculo, por lo que simplemente asignándole una variable ficticia que simbolizará "dificultad" puede ser suficiente. En el caso extremo, podría utilizarse una variante de los análisis de series temporales ponderadas, en el

cual, cada momento temporal incluye una variable ficticia (Green, 1993; Sayrs, 1989).

Estudios realizados por Kieras y Polson (Bovair, et al., 1990; Kieras y Bovair, 1986) utilizan un marco metodológico similar, aunque faltan dos elementos presentes en el análisis de series temporales ponderadas. En primer lugar, no hay control de la autocorrelación y en segundo lugar realizan un análisis equivalente a un modelo de efectos fijos frente a los modelos de efectos aleatorios utilizados en este trabajo. Aunque en nuestro caso, la autocorrelación encontrada en los estudios aquí mostrados no fue muy elevada, pensamos que la posibilidad de controlar todos estos efectos es de gran interés en este tipo de situaciones.

3) La variable dependiente tiempo para realizar la tarea

La variable dependiente utilizada en este estudio ha consistido en el tiempo que los usuarios necesitaban para completar la tarea en cada una de las ocasiones. Esta variable puede ser considerada como un compuesto de lo que en otros estudios serían consideradas como variables diferentes: tiempo de aprendizaje, tiempo de ejecución, tiempo de resolución de problemas, etc. Como hemos visto, siguiendo las predicciones realizadas por NGOMS, esta variable parece aproximarse bastante a lo predicho como tiempo de ejecución.

En este estudio, la variable dependiente utilizada ha tenido fundamentalmente una distribución no normal, con una asimetría positiva notable. Esto ha ocurrido tanto cuando se consideraban los errores como cuando el tiempo empleado en ellos era eliminado de los análisis. Esta distribución se ha repetido consistentemente a través de todos los niveles considerados.

Este resultado es completamente razonable. Ciertas acciones, debido a su novedad o su dificultad, producirán tiempos mucho más altos que las acciones que no plantean ningún problema al usuario. Lo contrario -acciones ejecutadas tan rápidamente que se convierten en valores extremos por debajo de la media- es muy difícil debido a que la media de ejecución no es excesivamente alta y queda un margen demasiado estrecho.

Esto nos ha llevado a realizar transformaciones logarítmicas (o similares) antes de intentar ajustar los modelos considerados para evitar las consecuencias negativas de la no-normalidad de estos datos. Ello significa, no

obstante, que las relaciones que encontraríamos entre las variables dependientes en la escala normal son no lineales (exponenciales).

Este es un resultado (encontrar relaciones no lineales entre el número de reglas y el tiempo de realización) un tanto inusual en la literatura, que puede ser explicado por las diferencias entre la situación planteada por nosotros respecto de las utilizadas por ejemplo en los estudios de Kieras y Polson (Bovair, et al., 1990; Kieras y Bovair, 1986) así como por las variables dependientes utilizadas. Esta diferencia consiste en, fundamentalmente, el haber proporcionado un número de instrucciones reducido de modo que los usuarios estuvieran obligados a realizar un mínimo de búsquedas por ensayo y error hasta lograr alcanzar la solución correcta en cada momento. Además, el sistema de ayuda en el que se mostraba la solución correcta en cada caso no produjo el efecto esperado en los sujetos, ya que, a menudo, los usuarios cometieron el mismo error repetidamente, a pesar de ver en la pantalla cual era la acción correcta a realizar después de haber cometido el error.

No obstante, a pesar de ese efecto aparentemente exponencial, ciertos predictores, tal como el número de reglas nuevas presentaron un comportamiento cuasilineal dentro de los valores máximos y mínimos considerados. Otros predictores por otro lado sí que tuvieron indudablemente una relación no lineal con la variable predicha.

4) Conclusiones por técnica utilizada.

1. NGOMS: Nivel 1

NGOMS ha producido resultados diferentes dependiendo del nivel de análisis considerado.

En el nivel superior de análisis se produjo una correlación entre el número de reglas nuevas y el orden en que los sujetos aprendieron las tareas que condujo a unos resultados nulos en los análisis de regresión. Además, la relación encontrada entre el número de reglas totales y las variables dependientes no fue excesivamente importante.

El haber encontrado una correlación importante entre el orden de la tarea y el número de reglas nuevas es importante, en nuestra opinión, porque limita la extensión que los análisis basados en NGOMS pueden tener en situaciones prácticas. Esta correlación es prácticamente inevitable que aparezca debido a que el número de reglas nuevas habitualmente descenderá a medida que ciertas tareas son aprendidas y, por tanto, ambos efectos son

difícilmente separables. No obstante, puede pensarse en situaciones experimentales en las que se busquen combinaciones de tareas en las que el número de reglas nuevas no esté relacionado con el orden de las tareas, pero, de acuerdo con la propia teoría, esto no es más que un interfaz mal construido ya que las ganancias de la consistencia en lugar de resultar positivas serían negativas. Una situación ciertamente indeseable.

Esto nos lleva a la paradoja de que para demostrar que este modelo funciona tenemos que construir malos interfaces intencionadamente. Una situación que, en el contexto de un trabajo práctico, no debería aparecer.

Un problema distinto es la falta de efecto de la variable TGOMS en el nivel superior. Al parecer, la longitud estimada de los métodos no es capaz de predecir la variable dependiente a este nivel, independientemente de cuestiones tales como el orden de la tarea o si ha sido realizada previamente o no. Esto nos lleva a pensar que las predicciones realizadas a nivel global, al no tener en cuenta la dinámica de las distintas tareas a nivel más concreto, no parecen ser capaces de realizar predicciones adecuadas una vez todos los efectos considerados han sido controlados.

2. NGOMS: Nivel 2

A este nivel, tanto NGOMS como TGOMS demostraron una relación con el tiempo necesario para realizar la tarea con respecto a las dos variables dependientes consideradas después de que estas fueran transformadas logarítmicamente. Además, al convertir los efectos de esas variables a la escala original se puede observar (figuras 5.12 y 5.19) que las puntuaciones predichas tienen una forma aproximadamente lineal dentro de los márgenes utilizados en este estudio para la variable dependiente NGOMS.

TGOMS en cambio parece presentar una forma curvilínea en su relación con las puntuaciones predichas en la escala original incluso sin realizar extrapolaciones.

La variable ficticia TAG (extraída a partir del análisis TAG y que indica consistencia v. inconsistencia) no produce un resultado satisfactorio. Aunque supuestamente debería producir el resultado de un tiempo superior a lo esperado, el resultado obtenido es el contrario, ya que los usuarios realizaron la tarea más rápidamente de lo predecible por el resto de la ecuación.

Estos resultados en su conjunto podrían considerarse en principio que respaldan el análisis NGOMS a este nivel, pero existen dos cuestiones que es

necesario considerar con más detalle antes de ofrecer un juicio definitivo. Estos son 1) la, en este caso, falta de correlación entre el orden de la tarea y el número de reglas nuevas, y, 2) el tamaño del efecto de los resultados encontrados.

1) La falta de correlación entre las reglas nuevas y el orden de la tarea en este caso se debe a que el nivel al que está definido el orden corresponde a un nivel superior al de las tareas consideradas por lo que para un mismo orden es posible encontrar diferentes valores de reglas a nivel de subtarea. Esto es esperable ya que la desagregación usualmente debería producir una disminución en la magnitud de la proporción de la varianza explicada entre dos variables. Así pues, parece razonable pensar que las predicciones basadas en NGOMS pueden ser más útiles a niveles de agregación intermedios ya que en ellos la correlación con el orden de la tarea en los datos resulta disminuida naturalmente y existe una mayor oportunidad de mostrar resultados interesantes.

2) El tamaño del efecto de los resultados encontrados es una cuestión de gran importancia a la hora de valorar el interés de estos. Efectos poco importantes del número de reglas nuevas podrían no justificar el esfuerzo de realizar modificaciones en el interfaz. Otro problema más grave existe también: tener que obtener las predicciones absolutas siempre a partir de estimaciones empíricas en lugar de a partir de valores determinados previamente. Empezaremos con el segundo problema en primer lugar:

Como es posible ver en el gráfico 6.17 y 6.18 del capítulo seis, el ajuste al nivel 2 entre los tiempos medios de los sujetos y las predicciones en tiempo de ejecución y tiempo de aprendizaje es visualmente satisfactorio. No obstante, si tenemos en cuenta que la escala para el caso del aprendizaje es distinta (minutos para las predicciones, segundos para los datos verdaderos) podemos ver como en realidad, en este caso, este ajuste no existe (gráfico 6.17) No obstante, para los tiempos de ejecución, una prueba t de comparaciones apareadas no encontró diferencias significativas entre los resultados hallados y las predicciones realizadas siguiendo las fórmulas de Kieras y Polson. Este es un resultado que sigue la misma línea de lo hallado mediante los análisis de series temporales, ya que aquéllos mostraban como el número de reglas totales era la variable independiente con un mayor efecto sobre el tiempo de resolución del problema.

La razón por la que la variable dependiente utilizada está más cargada del componente de ejecución que del componente de aprendizaje radica probablemente en que en la situación por la que pasaban los sujetos tuvieron la oportunidad de leer (y por tanto aprender a manejar) las tareas previamente a llevarlas a cabo efectivamente. Esta disposición pretendía simular la forma en la que muchos sujetos probablemente aprenden a manejar programas informáticos, primero leyendo las instrucciones escritas y luego intentando ejecutar las tareas sobre el mismo ordenador (aunque en nuestro caso no se les permitía repasar las instrucciones).

Ya que la predicción del tiempo de ejecución está determinada fundamentalmente por el número de reglas totales es razonable esperar que las predicciones de los tiempos medios ajusten adecuadamente.

3. TAG

Este método produjo efectos significativos a los niveles usuales, pero negativos, al predecir la variable tiempo sin error, y no produjo efectos significativos en los análisis de regresión.

Este resultado parece ser producto de la correlación entre esta variable ficticia y las variables extraidas a través del análisis NGOMS. Esta correlación transforma el efecto que esta variable tiene en principio sobre las variables dependientes (v. tabla 7.1) de tal modo que invierte la tendencia original.

LOGSIN	LOGCON	PRINTEN	INTENTO	ORDTAR	NGOMS	TGOMS	TAG
1.000							
0.861	1.000						
0.271	0.319	1.000					
-0.200	-0.234	-0.716	1.000				
0.049	-0.051	0.024	-0.075	1.000			
0.490	0.464	0.088	-0.077	0.070	1.000		
0.750	0.595	0.008	-0.020	0.229	0.296	1.000	
0.594	0.526	-0.002	0.004	-0.006	0.407	0.767	1.000

Tabla 7.1: Correlaciones entre las variables dependientes e independientes al nivel 2.

Este resultado puede ser interpretado como muestra de un grado de solapamiento entre los dos métodos para el análisis de la interacción a este nivel. Así, la correlación entre las reglas nuevas y TAG resulta esperable ya que ambos son métodos para el cálculo de la consistencia. Por otro lado, la correlación con el número de reglas totales es indicativo de una tendencia por parte de este método a considerar como inconsistentes las tareas más largas.

En definitiva, es posible concluir que existe suficiente solapamiento entre TAG y NGOMS como para que utilizar ambos procedimientos no sea interesante.

4. Action-Task.

Action Task es el análisis realizado a un nivel de grano más fino. Ello ha llevado naturalmente a producir una cantidad de datos mucho más grande y con una gran variabilidad por lo que las predicciones (desde el punto de vista de la proporción de varianza explicada) son los más bajos (alrededor de un 10%).

No obstante, la variable considerada obtiene unos resultados significativos a la hora de predecir el tiempo que los sujetos emplearán tanto para el caso en que se predice el tiempo sin errores como el tiempo con errores.

7.4. El valor de los resultados.

Los resultados obtenidos tal y como se discutió en parte inicial de este trabajo deberían alcanzar un tamaño de efecto suficientemente grande para realmente justificar su utilidad en situaciones aplicadas. La tabla 7.2 muestra los distintos tamaños del efecto para los métodos formales considerados obtenidos en este trabajo.

Nivel II		
Predicción	Tiempo sin error	Tiempo con error
NGOMS	.66 s. por regla	.77 s. por regla
TGOMS	Relación exponencial (en las 10 primeras reglas. 40 s. por regla aprx. pero incrementa más tarde)	Relación exponencial (en las 10 primeras reglas. 15 s. por regla aprx. pero incrementa más tarde)
Nivel III		
Action-Task	0.16 s por cada nueva opción.	0.09 s. por cada nueva opción

Tabla 7.2: Tamaños del efecto para las variables independientes predictoras con resultados significativos.

En la tabla superior se encuentran los valores encontrados para aquellas variables que tuvieron resultados estadísticamente significativos extraidas de los análisis formales. Con respecto al nivel II, como es posible ver, tres variables están relacionadas con las variables dependientes consideradas y ofrecen por tanto una oportunidad para mejorar el interfaz manejado. Como es posible, ver estas tres variables ofrecen mejoras de aproximadamente un segundo y medio por cada modificación con la excepción de TGOMS que ofrecería un efecto mucho más sustancial al ser la relación de

tipo exponencial. No obstante, este aumento, como vimos, está relacionado con la variable ficticia asignada a la tarea "Cambiar Comentario" por lo que resulta difícil extraer conclusiones.

En cualquier caso, el diseñador tendría la opción de proponer una serie de modificaciones para mejorar el interfaz en los aspectos considerados por las variables. ¿Cuáles deberían ser esas modificaciones? Dos consideraciones son importantes aquí: 1) lo fácilmente que la corrección puede ser realizada y 2) el impacto que la corrección tendrá sobre un uso continuo. Este impacto dependerá de dos cosas:

1) El grado de efecto que tenga el aprendizaje. Como vimos en el capítulo anterior, tanto el hecho de haber aprendido funciones anteriormente, como el haber realizado esa tarea significa una reducción en el tiempo de realización de las tareas. Llegado a cierto punto la reducción debería estabilizarse alcanzando un mínimo en el tiempo de realización. Por ello, si la tarea va a ser aprendida de manera muy intensa, y va a ser utilizada muy habitualmente, eliminar reglas, ya sean nuevas o totales, o aumentar la consistencia con el resto, puede no ser importante, porque la práctica que los usuarios obtengan producirá igualmente ese efecto.

No obstante, este efecto puede verse limitado por características del sistema especialmente mal planteadas, de tal modo que los usuarios no aprendan a realizar bien ciertas tareas a pesar de una gran cantidad de práctica y a que se equivoquen constantemente. En ese caso, los análisis cualitativos podrían revelar estas limitaciones. No obstante, en nuestro caso, los análisis formales utilizados no produjeron información importante de este tipo, por lo que no es posible ofrecer indicaciones claras al respecto. La única predicción cualitativa explorada (la tarea "Cambiar Comentario" está manifiestamente peor diseñada) produjo unos resultados contrarios a lo esperado.

2) El grado de utilización de la función. Si la función es poco utilizada, no habrá posibilidad de que se produzca el efecto de aprendizaje y cualquier modificación de las variables independientes produciría efectos muy beneficiosos. Por ello, la consistencia es seguramente más importante cuando las funciones son poco utilizadas que cuando lo son mucho, ya que el usuario deberá apoyarse en deducciones o en el conocimiento general del sistema al no poseer la práctica suficiente para actuar automáticamente.

Tanta la existencia de pocas reglas totales como la existencia de consistencia serán factores muy beneficiosos en esta situación.

Con Action-Task parece que el efecto es mucho menos importante que con las descripciones realizadas anteriormente. No obstante, hay que tener en cuenta que eliminar opciones no adecuadas a la situación en el interfaz puede ser muy fácil en ciertas ocasiones (aunque no en otras) y además sus efectos son muy duraderos ya que alcanzan a muchas situaciones. Por ello, aunque los coeficientes observados no son muy grandes en relación con las medianas de las acciones (mejores estimadores de la tendencia central en este caso) puede ser interesante realizar esta reducción.

7.5. Limitaciones de este estudio.

A continuación pueden encontrarse una serie de limitaciones de este estudio que trabajos posteriores deberían intentar superar.

Problemas para la demostración del valor de los métodos formales.

La utilidad de los métodos formales no es comprobable por medio de sólo un estudio. La siguiente discusión ofrece algunas sugerencias para estudios posteriores.

La prueba más importante para una herramienta de asistencia a la ingeniería es demostrar que es aceptada y utilizada en situaciones de diseño, además de contribuir positivamente a la mejora de los resultados obtenidos sin ella. Para demostrar esto es posible pensar en dos estrategias:

- La estrategia "natural" que consiste en observar el uso real de aquéllos que podrían sentir interés por estas técnicas en su medio ambiente de trabajo.

- La estrategia "artificial" que consistiría en utilizar grupos de diseñadores que, en las condiciones adecuadas, se esforzarán por utilizar las metodologías contrastando los resultados con, por ejemplo, un grupo que intentara hacer sus predicciones por medio de la intuición para así demostrar la bondad de las técnicas.

Ninguna de estas posibilidades resulta fácil debido a problemas prácticos. En las situaciones naturales, la industria ha sido bastante lenta (Monk, en prensa) en incorporar estas metodologías por lo que tampoco existen muchos informes acerca de estas cuestiones. La segunda estrategia

es poco viable también, ya que implicaría un grupo relativamente amplio de sujetos entrenados en el uso de una metodología, lo cual no está disponible fácilmente, para más tarde incorporar las sugerencias en prototipos diferentes y realizar análisis que demostrarían/no demostrarían el valor de las técnicas.

Hasta ahora, la mayoría de los trabajos presentados para validar métodos formales han utilizado el uso de ejemplos, bien construidos experimentalmente para tal fin o bien comerciales, analizados por el propio equipo encargado de la investigación. Sólo dentro del proyecto Amodeus (Shum, 1994), se manejó una situación en la que varios investigadores utilizaron diversas técnicas para realizar comparaciones entre ejemplos de diseño. No obstante, los prototipos diseñados no fueron más tarde construidos y analizados para comprobar si las afirmaciones encontradas eran empíricamente válidas.

En nuestro caso, un único diseñador ha realizado todos los análisis y ha construido el sistema que se deseaba analizar. Esto presenta el problema de que:

• Los resultados obtenidos son dependientes de la habilidad del único diseñador (el autor de este trabajo).

• Puede haber habido filtraciones de ideas o información de un análisis a otro.

Por otro lado, el tener un único diseñador puede presentar las siguientes ventajas sobre otro tipo de estudios:

• El investigador realmente ha intentado utilizar las técnicas para el objetivo propuesto. No se trata de sujetos con poco interés por lo que están haciendo.

• Algunas de las técnicas utilizadas necesitaban conocimientos y técnicas que un grupo en un experimento no está probablemente dispuesto a aprender. La solución podría ser un curso en el que la gente aprendería las técnicas y luego las emplearía en ejemplos de diseño. En el futuro esperamos tener la oportunidad de realizar este tipo de ejercicios.

Problemas para obtner la información

La información fue recogida a través de un programa informático que grabó la actividad de la pantalla mientras los sujetos realizaban las tareas analizadas. Estas grabaciones tenían la forma de archivos en el disco duro del ordenador las cuales posteriormente podían ser revisadas y etiquetadas por medio de otra aplicación. Estas etiquetas separaban bloques de acciones relevantes y proporcionaban su longitud temporal.

Debido a que el objetivo original del programa utilizado no es realizar análisis observacionales, las facilidades ofrecidas eran limitadas, por lo que cada análisis supuso una cantidad de tiempo muy considerable. El interés de este estudio en trabajar sobre el tiempo a lo largo de las diversas situaciones fue la causa de trabajar con análisis detallados de las actividades de los usuarios, en lugar de trabajar con tiempos agregados tal y como a menudo se ha realizado en estudios estudios mencionados en la literatura. Esto implica un estudio mucho más detallado de cada sujeto en lugar de utilizar medidas agregadas (tal y como la media del tiempo empleado por los sujetos para cada tarea) que no tendrían en cuenta el comportamiento en cada momento de los usuarios.

Debido a estas cuestiones, el número de sujetos no fue muy grande aunque la cantidad de información produjo una masa de datos relativamente considerable.

Hay que hacer notar que, en una situación aplicada en la que un sistema está en fase de análisis, no es probable que se dediquen muchos más recursos al análisis de un sistema que los aquí empleados. Por ello, comprobar que las técnicas de análisis formal son capaces de demostrar poder predictivo con un número reducido de sujetos es un objetivo interesante.

Falta demostrar que además de predecir es posible realizar sugerencias que mejoren el diseño.

Una de las cuestiones todavía por establecer en los métodos que se han utilizado en este trabajo es si, además de poder predecir el funcionamiento de los usuarios, es posible también plantear modificaciones sobre los sistemas que los conviertan en mejores y más fáciles de usar.

Para ello, las técnicas utilizadas deberían ofrecer un conjunto de recomendaciones y técnicas acerca de errores comunes en el diseño que afectan a las características de usabilidad de los sistemas así como métodos para corregir ese problema.

El análisis realizado con Action-Task puede ser un modelo de este tipo. Por ejemplo, dado que un número exagerado de acciones posibles es inadecuado, se pueden hacer las siguientes recomendaciones de diseño:

• Buscar que la situación refleje, teniendo en cuenta la historia de acciones pasadas, el menor número de acciones posibles.

• Utilizar menús y situaciones con el menor número de acciones posibles por cada decisión[1].

Estas sugerencias podrían ser proyectadas sobre las descripciones formales, de tal modo que luego se pudiera comprobar su posible efecto empírico. Al hacerlo, la utilidad de realizar descripciones formales recibiría un refuerzo importante.

Otra posibilidad es realizar comparaciones de sistemas con idéntica funcionalidad teniendo en cuenta las descripciones realizadas con las notaciones. Este tipo de estudios está en la actualidad llevándose a cabo.

Extender las predicciones a otras variables dependientes.

A pesar de que muchos métodos formales prometen ser capaces de realizar predicciones de otras variables dependientes además del tiempo de ejecución o aprendizaje, lo cierto es que en nuestro trabajo no hemos logrado llevar a la práctica las sugerencias propuestas. La carga mental por ejemplo, aunque utilizada en ciertos estudios basados en GOMS, no produjo en nuestro caso un análisis satisfactorio (prácticamente ninguna situación tenía una carga mental importante) y fue eliminada de los cálculos. Por otro lado, los errores, predecibles en principio mediante métodos basados en arquitecturas cognitivas, no pudieron tampoco ser anticipados por ninguno de los métodos.

---

[1] Esta recomendación no obstante puede aumentar el número de acciones necesarias para completar un objetivo por lo que puede no ser aceptable en ciertas ocasiones.

Otras variables, tal como la satisfacción del usuario, no son siquiera consideradas en la literatura acerca de métodos formales.

Valencia, Diciembre de 1995.

Referencias

Abowd, G. D. (1991). *Formal aspects of Human-Computer Interaction*. Trinity College. Oxford.

Agarwal, R. (1992). A Petri-Net based approach for verifying the integrity of production Systems. *International Journal of Man-Machine Studies*, **36**, 447-468.

Alexander, H. (1990). Structuring dialogues using CSP. In M. Harrison and H. Thimbleby (Eds.), *Formal Methods in Human-Computer Interaction.* (pp. 273-295). Cambridge: Cambridge University Press.

Alexander, P. (1987). Formally-based Methods for dialogue Design. In R. Diaper & W. Winder (Eds.), *People & Computers III.,*. Cambridge UK: Cambridge UP.

Apple (1987). *Human Interface Guidelines: The Apple Desktop Interface*. Reading, MA: Addison-Wesley Publishing Co.

Barnard, P., Wilson, M. & Maclean, A. (1987). Approximate modelling of Cognitive activity, towards an Expert System Design aid. In J. M. Carroll and P. Tanner (Ed.), *Human Factors in Computing Systems- and Graphics Interface*, (pp. 21-26). Toronto, Canada: North Holland.

Barnard, P. J. (1988). Cognitive resources and the learning of Human-Computer Dialogues. In J. M. Carroll (Eds.), *Interfacing thought. Cognitive aspects of Human-Computer Interaction*. Cambridge, Massachusetts: Bradford.

Barnard, P. J., MacLean, A. & Hammond, N. V. (1984). User representations of ordered sequences of Command Operations. In B. Shackel (Ed.), *Proceedings of Inter´84: First IFIP Conference on Human-Computer Interaction*, 1 (pp. 434-438). London.

Belloti, V. (1988). Implications of current Design practice for the use of HCI techniques. In D. M. Jones and R. Winder (Eds.), *People & Computers IV*. Cambridge: Cambridge University Press.

Blandford, A. & Young, R. (1993). *Developing Runnable User Models: Separating the Problem Solving Techniques from the Domain Knowledge* . AMODEUS Project.

Blandford, A. E. & Young, R. M. (1994). *PUMs Analysis of a prototype media space Design* (UM/WP24). Amodeus Project.

Bovair, S., Kieras, D. E. & Polson, P. G. (1990). The acquisition and performance of text-editing skill: A Cognitive Complexity Analysis. *Human Computer Interaction*, **5**, 1-48.

Browne, D. (1994). *STUDIO: STructured User-Interface Design for Interaction Optimisation.* Prentice Hall.

Byrne, M. D., Wood, S. D., Sukaviriya, P., Foley, J. D. & Kieras, D. E. (1994). Automating Interface Evaluation. In Dumais & Olson Adelson (Ed.), *CHI´94*, (pp. 232-237). Boston, Massachusetts: ACM.

Card, S. K., Moran, T. P. & Newell, A. (1983). *The psychology of human-computer Interaction.* New Jersey:

Carroll, J. M. (1982). Learning, using and designing Command paradigms. *Human Learning*, **1**, 31-62.

Carroll, J. M. & Rosson, M. B. (1991). Deliberated Evolution: Stalking the View Matcher in Design Space. *Human-Computer Interaction,* **6**(3,4), 281-318.

Carroll, J. M., Singley, M. K. & Rosson, M. B. (1992). Integrating Theory development with Design evaluation. *Behaviour and Information Technology*, **11**(5), 247-255.

Coutaz, J. (1987). PAC, an Object oriented Model for dialogue Design. In H. J. Bullinger and B. Shackel (Ed.), *Human-Computer Interaction INTERACT´87*, (pp. 431-436). Elsevier.

Coutaz, J., Nigay, L. & Salber, D. (1993a). *Conceptual software architecture Models for Interactive Systems* . Amodeus Project.

Coutaz, J., Nigay, L. & Salber, D. (1993b). *The MSM framework: A software Design space for multi-sensory-motor Interactive Systems* . Amodeus Project.

Cramer, M. (1990). Structure and mnemonics in computer and Command languages. *International Journal of Man-Machine Studies*, **32**, 707-722.

Curry, M. B. & Monk, A. F. (1990a). *Deriving and Representing Application Models: Dialogue Modelling and Analysis* . DREAM project.

Curry, M. B. & Monk, A. F. (1990b). *Dialogue Modelling and Analysis* . Data Logic.

Curry, M. B., Monk, A. F. & Maidment, B. A. (en prensa). Task-Based Interface Specification.

de Haan, G., van der Veer, G. C. & van Vliet, J. C. (1991). Formal modelling techniques in human-computer Interaction. *Acta Psychologica*, **78**, 27-67.

Dix, A. (1991). *Formal Methods for interactive systems.* Academic Press.

Dix, A., Finlay, J., Abowd, G. & Beale, R. (1993). *Human-Computer Interaction*. Prentice Hall.

Elkerton, J. & Palmiter, S. L. (1991). Designing help using a GOMS Model: An information retrieval evaluation. *Human Factors*, **33**(2), 185-204.

Faconti. (1993). *Towards the concept of interactor* . Amodeus Project.

Farooq, M. U. &. D., W. D. (1988). A survey of formal tools and Models for developing User interfaces. *International Journal of Man-Machine Studies.*, **29**(5), 479-496.

Firth, C. & Thomas, R. C. (1991). The use of Command Grammar in a Design tool. *International Journal of Man-Machine Studies*, (34), 479-496.

Gaines, B. R. & Shaw, M. L. G. (1986). From timesharing to the sixth generation: the development of human-computer Interaction. Part I. *International Journal of Man-Machine Studies*, **24**, 1-27.

Gifi, A. (1990). *Nonlinear Multivariate Analysis.* Chichester: Wiley and Sons.

Gong, R. & Kieras, D. (1994). A validation of the GOMS Model methodology in the development of a specialised, commercial software application. In B. Adelson, S. Dumais and J. Olson (Ed.), *Human Factors in Computing Systems*, (pp. 351-357). Boston, Massachusetts: CAI.

Gould, J. D. (1987). How to Design Usable Systems. In *Proceedings of IFIP INTERACT'87: Human-Computer Interaction*. (pp. xxxv-xli).

Gray, W. D., John, B. E. & Atwood, M. E. (1993). Project Ernestine: Validating a GOMS Analysis for predicting and explaining real-world Task performance. *Human-Computer Interaction*, **8**(3), 237-309.

Gray, W. D., John, B. E., Stuart, R., Lawrence, D. & Atwood, M. E. (1990). GOMS Meets the Phone Company: Analytic Modelling Applied to Real-World Problems. In *Proceedings of IFIP INTERACT'90: Human-Computer Interaction*. (pp. 29-34).

Green, T. R. G. & Payne, S. J. (1984). Organisation and learnability in computer languages. *International Journal of Man-Machine Studies*, **21**, 7-18.

Green, T. R. G., Schiele, F. & Payne, S. J. (1988). Formalisable Models of User knowledge in human-computer Interaction. In G. Van der Veer, T. R. G. Green, J. M. Hoc and D. Murray (Eds.), *Working with computers: Theory versus outcome*. Academic Press.

Green, W. H. (1993). *Econometric Analysis*. MacMIllan.

Greenacre, M. J. (1989). *Theory and applications of correspondence Analysis*. San Diego: Academic Press.

Gugerty, L. (1993). The use of analytical Models in human-computer interface Design. *International Journal of Man-Machine Studies*, (38), 625-660.

Harrison, M. & Abowd, G. D. (1991). *Formal Methods in Human-Computer Interaction: A tutorial* . Human Computer Interaction Group. University of York.

Harrison, M. D. & Timbleby, H. (Ed.). (1990a). *Formal Methods in Human-Computer Interaction*. Cambridge: Cambridge University Press.

Harrison, M. D. & Timbleby, H. (1990b). The role of Formal Methods in Human-Computer Interaction. In M. D. Harrison and H. Timbleby (Eds.), *Formal*

Methods in Human-Computer Interaction. Cambridge: Cambridge University Press.

Hayes-Roth, F. & Jacobstein, N. (1994). The State of Knowledge-Based Systems. *Communications of the ACM*, **37**(3), 27-39.

Heid, J. & Norton, P. (1989). *Inside the Apple Macintosh*. Simon & Schuster.

Hoaglin, D. C., Mosteller, F. & Tukey, J. W. (Ed.). (1983). *Understanding robust and exploratory data Analysis*. New York: Wiley.

Howes, A. (1994). A Model of the acquisition of menu knowledge by exploration. In B. Adelson, S. Dumais and J. Olson (Ed.), *CHI´94*, (pp. 445-451). Boston, Massachusetts: ACM.

Howes, A. & Payne, S. J. (1990). Display-based competence: Towards User Models for menu-driven interfaces. *International Journal of Man-Machine Studies*, **33**(6), 637-655.

Howes, A. & Young, R. M. (1991). Predicting the Learnability of Task-Action Mappings. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*. (pp. 113-118).

Irving, S., Polson, P. & Irving, J. E. (1994). A GOMS Analysis of the advanced automated cockpit. In B. Adelson, S. Dumais and J. Olson (Ed.), *Human Factors in Computing Systems*, (pp. 344-350). Boston, Massachusetts: CAI.

Jacob, R. J. K. (1983). Using formal specifications in the Design of a Human-computer Interface. *Communications of the ACM*, **26**(4), .

Jacob, R. J. K. (1986). A Specification Language for direct manipulation. *ACM Transactions on Graphics*, **4**(4), 283-317.

Johnson, C. W. & Harrison, M. D. (1990). *Complexity, Consistency and Control: A formal critique of Object oriented Interaction* . The Human Computer Interaction Group, The dept. of Computer Science, The University of York.

Johnson, P. (1992). *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering*. McGraw Hill.

Karat, J. & Bennett, J. (1991). Modelling the User Interaction Methods imposed by designs. In M. J. Tauber and D. Ackerman (Eds.), *Mental Models and Human Computer Interaction 2*. (pp. 257-269). Elsevier Science Publishers.

Kieras, D. (1988). Towards a practical GOMS Model methodology for User interface Design. In H. Helander (Eds.), *Handbook of Human-Computer Interaction*. (pp. 135-157). New York: North Holland.

Kieras, D. (1991). *A Guide to GOMS Task Analysis* . University of Michigan.

Kieras, D. & Polson, P. G. (1985). An approach to the formal Analysis of User Complexity. *International Journal of Man-Machine Studies*, **22**, 365-394.

Kieras, D. E. & Bovair, S. (1986). The acquisition of procedures from text: A production-System Analysis of transfer of training. *Journal of Memory and Language*, **25**, 507-524.

Kirakowski, J. & Corbett, M. (1990). *Effective Methodology for the study of HCI*. North-Holland.

Kleinbaum, D. G., Kupper, L. L. & Muller, K. E. (1988). *Applied Regression Methods and other multivariable Methods*. Boston: PWS-Kent Publishing Company.

Laird, J., Newell, A. & Rosenbloom, P. (1987). An architecture for general intelligence. *Artificial Intelligence*, (33), 1-64.

Laird, J. E., Congdom, C. B., Altmann, E. & Doorenbos, R. (1993). *Soar User´s Manual* . Technical Report Electrical Engineering and Computer Science Department & School of Computer Science Carnegie Mellon University.

Landauer, T. K. (1987). Relations between Cognitive Psychology and Computer System Design. In J. M. Carroll (Eds.), *Interfacing Thought: Cognitive aspects of human-computer Interaction*. MIT Press.

Lerch, F. J., Mantei, M. M. & Olson, J. R. (1989). Skilled Financial Planning: The Cost of Translating Ideas into Action. In *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*. (pp. 121-126).

Lewis, C., Casner, S., Schoenberg, V. & Blake, M. (1987). Analysis-Based Learning in Human-Computer Interaction. In *Proceedings of IFIP INTERACT'87: Human-Computer Interaction*. (pp. 275-280).

Long, J. & Whitefield, A. (Ed.). (1989). *Cognitive Ergonomics and Human-Computer Interaction*. Cambridge, UK: Cambridge University Press.

Maddala, G. S. (1971). The use of variance components Models in pooling cross section and time series data. *Econometrica*, **29**(2), 341-358.

Maddala, G. S. (1985). *Econometría*. MacGraw Hill.

Maddala, G. S. (1987a). Limited dependent variable Models using panel data. *The Journal of Human Resources*, **22**, 307-338.

Maddala, G. S. (1987b). Recent developments in the econometrics of panel data Analysis. *Transportation Research*, **21**(4/4), 303-326.

May, J., Barnard, P. J. & Blandford, A. (1992). *Using Structural Descriptions of Interfaces to Automate the Modelling of User Cognition* (UM/WP1). Amodeus Project.

May, J., Tweedie, L. A. & Barnard, P. J. (1993). *Modelling User performance in visually based interactions* (UM/WP3.2). Amodeus Project.

Mayes, J. T., Draper, S. W., McGregor, M. A. & Oatley, K. (1988). Information flow in a User interface: the effect of experience and context on the recall of MacWrite screens. In D. M. Jones and R. Winder (Eds.), *People and Computers IV*. Cambridge, UK: Cambridge University Press.

Monk, A. (1990). Action-Effect rules: a technique for evaluating an informal Specification against principles. *Behaviour and Information Technology*, **9**(2), 147-155.

Monk, A. (1993). *Deriving and representing application Models* . Report on the collaborative project between York University, Data Logic Ltd. Harrow and System Concepts Ltd.

Monk, A., Curry, M. & Wright, P. (en prensa). Why industry doesn't use the wonderful notations we researchers have given them to reason about their designs. In D. J. Gilmore (Eds.), *User-centred requirements for software engineering*. Springer-Verlag.

Monk, A., Nardi, B., Gilbert, N., Mantei, M. & McCarthy, J. (1993). Mixing Oil and Water? Ethnography versus Experimental Psychology in the Study of Computer-Mediated Communication. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems.* (pp. 3-6).

Monk, A. F. & Whright, P. C. (1991). Observations and inventions: new approaches to the study of human-computer Interaction. *Interacting with computers*, **3**(2), 204-216.

Moore, K. J., Osgood, D. W., Larzelere, R. E. & Chamberlain, P. (1994). Use of pooled time series in the study of naturally ocurring clinical events and problem behavior in a foster care setting. *Journal of Consulting and Clinical Psychology*, **62**(4), 718-728.

Moran, T. (1981a). An applied psychology of the User. *Computing surveys*, **13**(1), 1-11.

Moran, T. P. (1981b). The Command Language Grammar: a Representation for the User interface of Interactive computer Systems. *International Journal of Man-Machine Studies*, (15), 3-50.

Myers, K. J. & Hammond, N. V. (1991). *Consolidate report of workshop on scenario matrix Analysis* . ESPRIT 3066 Amodeus Deliverable D9. Dept. of Psychology, Univ. of York, UK.

Neves, D. M. & Anderson, J. R. (1981). Knowledge compilation: Mechanisms for the automatization of Cognitive skills. In J. R. Anderson (Eds.), *Cognitive skills and their acquisition*. Lawrence Erlbaum Associates.

Newell, A. & Card, S. K. (1985). The Prospects for Psychological Science in Human-Computer Interaction. *Human-Computer Interaction*, **1**(3), 209-242.

Newell, A. & Simon, H. (1972). *Human Problem Solving*. Englewood Cliffs. NJ: Prentice-Hall.

Nigay, L., Coutaz, J. & Salber, D. (1993). *A multimodal airline travel information System* . Amodeus Project.

Norman, D. A. (1981). The trouble with UNIX. *Datamation, ,* 139-150.

Norman, D. A. (1983a). Design rules based on Analysis of human error. *Communications of the ACM*, **4**, 254-258.

Norman, D. A. (1983b). Some Observations on Mental Models. In D. and Stevens Gentner A. L. (Eds.), *Mental Models*. (pp. 7-14). Hillsdale, NJ: Lawrence Erlbaum Associates.

Norman, D. A. (1984). Stages and levels in human-machine interaction. *International Journal of Man-Machine Studies*, **21**, 365-375.

Norman, D. A. (1986). Cognitive Engineering. In D. A. Norman and S. W. Drapper (Eds.), *User centered System Design*. (pp. 33-65). New Jersey: Lawrence Erlbaum Associates.

Norman, D. A. (1987). Cognitive engineering-Cognitive Science. In J. M. Carroll (Eds.), *Interfacing Thought: Cognitive aspects of Human-Computer Interaction*. (pp. 325-336). Cambridge: A Bradford Book.

Norman, D. A. (1988). *La Psicología de los objetos cotidianos*. New York: Basic Books.

Norman, D. A. & Draper, S. W. (Ed.). (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Novales Cinca, A. (1988). *Econometría*. MacGraw-Hill.

Olsen, D. R., Monk, A. & Curry, M. B. (1995). Algorithms for automatic dialogue Analysis using prepositional production Systems. *Human Computer Interaction*, **10**(1), 39-78.

Olson, J. R. & Olson, G. M. (1990). The Growth of Cognitive Modelling in Human-Computer Interaction Since GOMS. *Human-Computer Interaction*, **5**(2-3), 221-265.

Ostrom, C. W. (1978). *Time Series Analysis: Regression Techniques*. Beverly Hills, CA: Sage.

Paterno, F. (1993). *A methodology to Design Interactive Systems based on interacts* . Amodeus Project.

Paterno, F. (1994). A formal approach to the evaluation of Interactive Systems. *SIGCHI bulletin*, **26**(2), 69-73.

Payne, S. J. (1988). Metaphorical Instruction and the early learning of an abbreviated-Command computer System. *Acta Psychologica*, **69**, 207-230.

Payne, S. J. & Green, T. R. G. (1986). Task-Action Grammars: A Model of the mental Representation of Task languages. *Human Computer Interaction*, **2**(2), 93-133.

Pedhazur, E. J. (1982). *Multiple regression in behavioral research*. New York: Holt, Rinehart & Winston.

Peterson, J. L. (1977). Petri nets. *ACM Computing Surveys*, **9**(223-252), .

Pfaff, C. E. (Ed.). (1985). *User Interface Management Systems*. Berlin: Springer Verlag.

Pinillos, J. L. (1981). Observaciones sobre la Psicología científica. In V. Pelechano y otros (Eds.), *Psicologema*. (pp. 27-76). Valencia: Alfaplus.

Polson, P. G. (1987). A quantitative Theory of human-computer Interaction. In J. M. Carroll (Eds.), *Interfacing Thought: Cognitive aspects of human-computer Interaction*. MIT Press.

Polson, P. G. & Lewis, C. H. (1990). Theory-Based Design for Easily Learned Interfaces. *Human-Computer Interaction*, **5**(2-3), 191-220.

Reisner, P. (1981). Formal Grammar and Human Factors Design of an Interactive Graphics System. *IEEE Transactions on Software Engineering*, **7**(2), 229-240.

Reisner, P. (1990). What is inconsistency? In Diaper et. al (Ed.), *Proceedings of IFIP INTERACT'90: Human-Computer Interaction*, (pp. 175-181).

Reisner, P. (1993). APT: a description of User interface inconsistency. *International Journal of Man-Machine Studies*, (39), 215-236.

Richards, J. N. J., Bez, H. E., Gittins, D. T. & Cooke, D. J. (1986). On Methods for interface Specification and Design. *International Journal of Man-Machine Studies*, (24), 545-568.

Ritter, F. & Larkin, J. H. (1994). Developing process Models as summaries of HCI Action sequences. *Human Computer Interaction*, **9**(3&4), 345-384.

Sayrs, L. (1989). *Pooled time series Analysis*. Beverly Hills: Sage Publicationes.

Schiele, F. & Green, T. R. G. (1990). HCI formalisms and Cognitive psychology: the case of Task-Action Grammar. In M. D. Harrison and H. Timbleby (Eds.), *Formal Methods in Human-Computer Interaction*. Cambridge: Cambridge University Press.

Serius (1992). *Serius Development System*. Salt Lake City: Serius.

Sharrat, B. (1987a). The incorporation of early interface evaluation into Command Language Grammar specifications. In *People and Computers III, Proceedings of Human-Computer Interaction 87, CUP*.

Sharrat, B. (1987b). Top-Down Interactive Systems Design: Some lessons learnt from using Command Language Grammar. In *INTERACT'87*, (pp. 395-399).

Shneiderman (1982). Multiparty Grammars and related features for defining Interactive Systems. *IEEE Transactions on Systems, Man & Cybernetics*, **12**(2).

Shneiderman, B. (1986). *Designing the User interface.* Addison-Wesley.

Shum, S., Jorgensen, A., Hammond, N. y Aboulafia, A. (1994). *Amodeus-2 HCI Modelling and Design Approaches: Executive Summaries and Worked examples* . Amodeus.

Shum, S. & Hammond, N. (1993). *Analysis of the Expert System Modeller as a Vehicle for ICS Encapsulation* (TA/WP5). Amodeus Project.

Simon (1981). *The sciences of the artificial*. Cambridge. MA: MIT Press.

Simon, T. (1988). Analysing the scope of Cognitive Models in human-computer Interaction: A trade-off approach. In D. M. Jones and R. Winder (Eds.), *People & Computers IV*. Cambridge: Cambridge University Press.

Smith, D. C., Irby, C., Kimball, R., Verplank, W. L. & Harslem, E. (1982). Designing the Star User Interface. *Byte*, **7**(4), 653-661.

Smith, S. L. & Mosier, J. N. (1986). *Guidelines for designing User interface software*. Bedford Massachusetts: Approved for public release.

Valero Mora, P. M., Molina Ibáñez, J. G., Sanmartín Arce, J. & Canet Centelles, F. (1994). Steps Towards a delimitation of HCI. In *23rd International Congress of Applied Psychology*, . Madrid, Spain:

Valero, P., Sanmartín, J. & Molina, G. (En preparación). Una Aproximacion al Uso de iconos en los Interfaces de Ordenador.

Van Biljon, W. R. (1988). Extending Petri nets for specifying man-machine dialogues. *International Journal of Man-Machine Studies.*, **28**(4), 437-455.

Wandmacher, J. & Arend, U. (1985). Superiority of global features in classification and matching. *Psychological Research*, **47**, 143-157.

Wasserman, A. I. (1986). Extending State Transition diagrams for the Specification of human-computer Interaction. *IEEE Transactions on Software Engineering*, **SE-11**, 147-156.

Whiteside, J., Bennett, J. & Holtzblatt, K. (1988). Usability Engineering: Our Experience and Evolution. In Martin Helander (Eds.), *Handbook of Human-Computer Interaction*. (pp. 791-817). New York, NY: North-Holland.

Whiteside, J. & Wixon, D. (1987). Discussion: improving human-computer Interaction-a quest for Cognitive science. In J. M. Carroll (Eds.), *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. (pp. 353-365). Cambridge MA: MIT Press.

Wilkinson, L. (1989). *Systat: The System for statistics*. Evanston, Illinois.: Systat Inc.

Williges, R. C. (1987). The use of Models in human-computer interface Design. *Ergonomics*, **30**(3), 491-502.

Woods, D. D. & Roth, E. M. (1988). Cognitive Systems Engineering. In Martin Helander (Eds.), *Handbook of Human-Computer Interaction*. (pp. 3-43). New York, NY: North-Holland.

Young, R. M. & Abowd, G. D. (1994). Multi-perspective modelling of interface Design issues: Undo in a collaborative editor. *Manuscrito no publicado.*

Young, R. M., Green, T. R. G. & Simon, T. (1989). Programmable User Models for Predictive Evaluation of Interface Designs. In *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*, (pp. 15-19).

Young, R. M., Howes, A. & Whittington, J. (1990). A knowledge Analysis of interactively. In Diaper et al. (Ed.), *INTERACT´90*, (pp. 115-120). Elsevier.

Young, R. M. & Whittington, J. (1990). Using a knowledge Analysis to predict conceptual errors in text-editor usage. In J. C. & Whiteside Chew J. (Ed.), *Proceedings of CHI´90 Human Factors in Computing Systems*, (pp. 91-97). ACM Press.

Anexo I

Manual de entrenamiento

M A N U A L

D E

E N T R E N A M I E N T O

E N

P R O G R A M A M I N I M O

INTRODUCCIÓN.

En este manual se explicará como manejar un programa sencillo de ordenador, que permite llevar a cabo manipulaciones de textos sencillos, tales como cambiar el tipo de letra, el tamaño, imprimirlo en papel o sacar una copia.

La forma en que está planteado el manual es la siguiente: En primer lugar se indica en letra grande y al comienzo de página el nombre de la función a explicar. A continuación se da una explicación de esa función y se añade un gráfico indicando los pasos principales necesarios para llevarla a cabo. Cada uno de estos pasos está indicado también en letra grande, explicado en una pequeña introducción y acompañado de los subpasos (pasos más concretos) que son necesarios para llevarlo a cabo.

Estudia este manual durante el tiempo que te marque el encargado del experimento. Intenta asimilar la mayor cantidad de información, pero ten en cuenta que el programa que vas a aprender está preparado para darte información si la necesitas.

FUNCIONES DEL PROGRAMA.

1. CAMBIAR EL TIPO Y EL TAMAÑO DEL TEXTO.

Esta función permite cambiar el tipo de letra con el que está escrito un texto (por ejemplo, de este Tipo DE LETRA a este otro TIPO DE LETRA) o el tamaño (por ejemplo de este tamaño a este otro tamaño.)

Los cuatro tipos de letras que tiene nuestro programa son así como una muestra del aspecto que tienen son los siguientes.

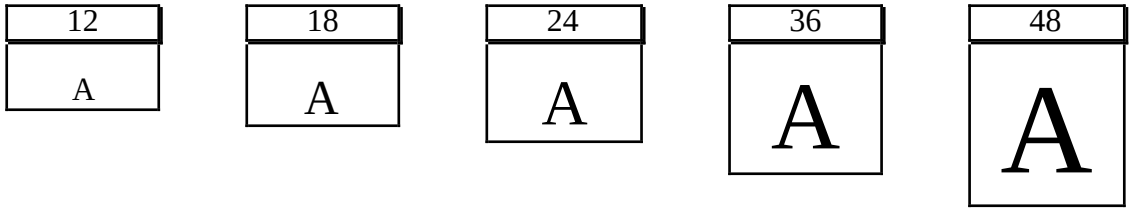
• Chicago: abcdefghijklmn opqrstuvwxyzABCDEFGHIJKLMN OP QRSTUVWXYZ

• Nueva York: abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOP QRSTUVWXYZ

• Geneva: abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTU VWXYZ.

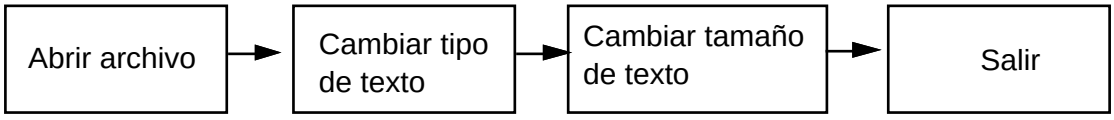
• Times: abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ

Los tamaños de letra está numerados y son posibles los siguientes: 12, 18, 24, 36, 48. A continuación se muestran estos tamaños aplicados al tipo de letra Times.



En nuestro sistema es necesario llevar a cabo los cambios de tipo de texto antes que los de tamaño, y llevar a cabo siempre ambos.

Los pasos a seguir para cambiar el tipo y el tamaño del texto son los que se muestran en el diagrama siguiente. Estos pasos se explican en más detalle a continuación.



1. ABRIR ARCHIVO.

Abrir archivo sirve para mostrar el texto sobre el que vamos a llevar a cabo las modificaciones.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Abrir.

3) Seleccionar el *archivo a abrir*[1].

4) Apretar el botón OK .

2. CAMBIAR TIPO DE LETRA DEL TEXTO.

Cambiar tipo de letra cambia el tipo de letra del texto que queremos modificar.

1) Descender el menú Tipo.

2) Soltar el botón del ratón en el comando *Tipo de letra nuevo*[2]

3. CAMBIAR TAMAÑO DE TEXTO.

Cambiar tamaño de letra cambia el tamaño de letra del texto que queremos modificar.

1) Descender el menú Tamaño.

2) Soltar el botón del ratón en el comando *Tamaño de letra nuevo*.

4. SALIR.

Salir termina con el ejercicio que estemos llevando a cabo.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Salir.

---

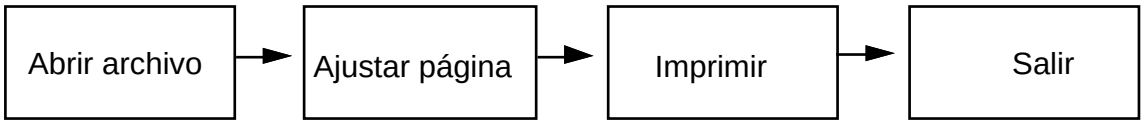
[1] *Archivo a abrir* está en cursiva porque el nombre del archivo puede variar de una ocasión a otra. Otros valores que también pueden variar de una ocasión a otra estarán más adelante escritos en cursiva.

[2] *Tipo de letra nuevo* está en cursiva por la misma razón que *Archivo a abrir*.

2. IMPRIMIR UN TEXTO.

Imprimir un texto manda un texto a la impresora para sacar una copia en papel. Para llevar a cabo esa impresión es necesario tener en cuenta ciertos detalles tales como el tamaño del papel o la orientación de la página (apaisada, normal, etc.)

Los pasos a seguir para imprimir el texto son los que se muestran en el diagrama siguiente. Estos pasos se explican en más detalle a continuación.



1. ABRIR ARCHIVO.

Abrir archivo sirve para mostrar el texto sobre el que vamos a llevar a cabo las modificaciones.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Abrir.

3) Seleccionar el *archivo a abrir*.

4) Apretar el botón OK .

2. AJUSTAR PÁGINA.

Ajustar página permite obtener modificar ciertos valores en relación con la impresora que estamos utilizando, tales como el tamaño de la página, su orientación (apaisada o normal), etc.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Ajustar página...

3) Apretar el botón OK.

3. IMPRIMIR.

Imprimir permite sacar una copia en papel del texto que está abierto en ese momento.

1) **Descender el menú Archivo.**

2) **Soltar el botón del ratón en el comando Imprimir…**

3) **Apretar el botón OK.**

4. SALIR.

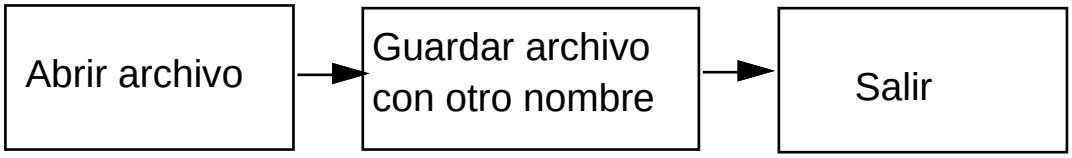
Salir termina con el ejercicio que estemos llevando a cabo.

1) **Descender el menú Archivo.**

2) **Soltar el botón del ratón en el comando Salir.**

3. SACAR UNA COPIA DE UN TEXTO.

Sacar una copia de un texto hace una copia del texto en el disco interno. Los pasos a seguir para sacar una copia del texto son los que se muestran en el diagrama siguiente. Estos pasos se explican en más detalle a continuación.



1. ABRIR ARCHIVO.

Abrir archivo sirve para mostrar el texto sobre el que vamos a llevar a cabo las modificaciones.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Abrir.

3) Seleccionar el *archivo a abrir*.

4) Apretar el botón OK .

2. GUARDAR ARCHIVO CON OTRO NOMBRE.

Guardar archivo con otro nombre permite sacar la copia del archivo que está abierto en ese momento.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Guardar Como….

3) Escribir nuevo nombre usando el teclado.

4) Hacer click en el botón OK.

3. SALIR.

Salir termina con el ejercicio que estemos llevando a cabo.

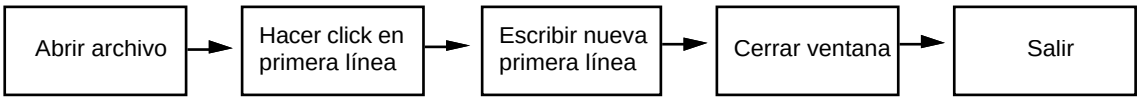
1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Salir.

4. CAMBIAR LA PRIMERA LíNEA POR DEFECTO.

Cambiar la primera línea permite modificar una línea que aparece en todos los textos. Esa línea debido a que aparece automáticamente puede ahorrarnos trabajo al no tener que escribirla cada vez. En otras ocasiones, no obstante, no será de utilidad y será necesario quitarla o cambiarla por otra más apropiada. En otras ocasiones, esta línea simplemente no aparecerá.

Los pasos para cambiar la primera línea por defecto son:



1. ABRIR ARCHIVO.

Abrir archivo sirve para mostrar el texto sobre el que vamos a llevar a cabo las modificaciones.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Abrir.

3) Seleccionar el *archivo a abrir*.

4) Apretar el botón OK .

2. HACER CLICK EN PRIMERA LÍNEA.

Hacer click en primera línea muestra la primera línea por defecto.

1) Mover la flecha del ratón hasta la primera línea del texto.

2) Apretar el botón del ratón.

3. ESCRIBIR NUEVA PRIMERA LÍNEA.

En este paso se escribe el texto de la primera línea.

1) Escribir la nueva primera línea usando el teclado

4. CERRAR VENTANA.

En este paso se cierra la ventana en la que se cambia la primera línea.

1) Mover la flecha del ratón hasta el cuadrado blanco de la izquierda.

2) Apretar el botón del ratón.

5. SALIR.

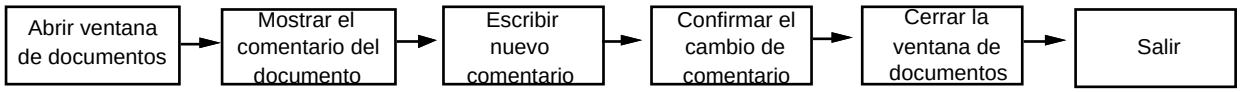
Salir termina con el ejercicio que estemos llevando a cabo.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Salir.

5. CAMBIAR UN COMENTARIO.

Los comentarios serían pequeñas notas que los usuarios podrían añadir a los textos para recordar detalles sobre estos. Esta función permitiría añadir o modificar estos comentarios.



1. ABRIR VENTANA DE DOCUMENTOS.

Abrir archivo sirve para mostrar el texto sobre el que vamos a llevar a cabo las modificaciones.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Abrir.

3) Seleccionar el *archivo a abrir*.

4) Apretar el botón OK .

2. MOSTRAR EL COMENTARIO DEL DOCUMENTO.

Mostrar el comentario del documento permite ver el comentario que tiene un documento dado, y, en su caso, modificarlo.

1) Hacer click en el documento que quieres cambiar.

2) Hacer click en el botón CAMBIAR COMENTARIOS.

3. ESCRIBIR NUEVO COMENTARIO.

Escribir nuevo comentario permite añadir o modificar un comentario a un documento.

1) Escribir el nuevo comentario usando el teclado.

4. CONFIRMAR EL CAMBIO DE COMENTARIO.

Esta acción confirma que el cambio del comentario está ya realizado y cierra la ventana en donde esto se lleva a cabo.

1) Hacer click sobre el botón de OK.

5. CERRAR LA VENTANA DE DOCUMENTOS.

Este paso cierra la lista en la que aparecen los documentos.

1) Hacer click sobre el botón Hecho.

6. SALIR.

Salir termina con el ejercicio que estemos llevando a cabo.

1) Descender el menú Archivo.

2) Soltar el botón del ratón en el comando Salir.

Anexo II
Recuentos en NGOMS

Reglas totales

Para llevar a cabo un recuento por medio de NGOMS es necesario empezar por el último nivel de la jerarquía. En este nivel, cada regla cuenta con un valor de uno por lo que, por ejemplo, el método para seleccionar de lista (v. figura I.1) necesitaría de seis pasos. No obstante, como la sentencia de comienzo del método cuenta como una regla, la longitud total de éste método se considera como igual a siete.

```
Método para Seleccionar de lista <item de lista>
1. Recuperar <item de lista> y Mirar instrucciones <item de lista> y Registrar <item de lista>
2. Recuperar <item de lista> y Localizar <item de lista> y Registrar <posición %%>
3. Recuperar <posición %%> y Mover flecha-ratón a <posición %%>
4. Clickar
5. Recuperar <item de lista> y Verificar resultado <item correcto seleccionado>.
6. Olvidar <item de lista> y Olvidar <posición %%> y Devolver Objetivo realizado.
```

Figura I1: Método para seleccionar de lista.

Para contar la longitud de un método a nivel 2 sumamos en primer lugar la longitud de los submétodos que son llamados por éste. Por ejemplo, en la figura I2 podemos ver los pasos de los que se compone la operación de abrir archivo entre parentesis. El resultado es 21.

```
1.      Abrir archivo
```

```
1. Retener que el comando es Abrir y Seleccionar
comando menu <comando> (8)
2. Verificar resultado <Ventana abrir abierta>
3. Retener que el item de lista es <nombre de
fichero> y  Seleccionar de lista <item de lista>
(7)
4. Retener que Tipo de botón es Confirmar y
Presionar botón <tipo de botón> (6)
5. Verificar resultado <documento abierto>.
6. Devolver Objetivo realizado.
```

Figura I2: Método para Abrir Archivo.

A este valor es necesario sumar el número de pasos de que se compone el propio método de Abrir Archivo (6) más la propia sentencia de método (1). Esto produce un valor de longitud total de 31.

Por último, para calcular la longitud total un nivel más arriba es necesario realizar cálculos similares con los métodos definidos en él. Por ejemplo, el método para Imprimir es el mostrado en la figura I3. La suma de la longitud de los submétodos es 60. A esto hay que añadirle cinco por la longitud del propio método y uno por la llamada a este método. En total 66.

```
1.      Método para Imprimir
1.      Abrir archivo (28)
2.      Ajustar Página (20)
3.      Imprimir (20)
4.      Salir (12)
5.      Devolver Objetivo realizado.
```

Figura I3: Método para Imprimir.

Reglas nuevas

El número de reglas nuevas en cada nivel es calculado restando las reglas que corresponden a reglas ya aprendidas anteriormente a ese nivel. Un procedimiento alternativo, en el que las reglas verdaderamente nuevas independientemente del nivel eran las únicas consideradas, fue rechazado después de mucha deliberación. Este último procedimiento producía que, por ejemplo, las reglas asociadas a bajar un menú, no fueran consideradas ya que habían sido ensayadas previamente y los usuarios eran conocedores del procedimiento. Del mismo modo, al nivel superior analizado (tal como por ejemplo la tarea Imprimir compuesta por Abrir archivo, ajustar página, etc.) buena parte de las subrutinas incluidas dentro de él habían sido practicadas anteriormente y sólo en ocasiones el cálculo de la consistencia a ese nivel hacía justicia a esas similaridades (por ejemplo, todos los métodos coincidían en el último paso-salir- pero eso consistencia no siempre era captada por el método).

El método seguido aquí parece ajustarse más a la propuesta realizada por Kieras et. al. por lo que ha sido el finalmente utilizado. Otros estudios podrían plantear la comparación entre estos dos métodos.

En detalle, el método utilizado implica lo siguiente:

• Si un método ha sido practicado con anterioridad el número de reglas nuevas que posee es cero (p.e abrir archivo sólo tiene reglas nuevas la primera vez que es utilizado).

• Si un método X presenta consistencia total con el método Y, y el método Y ha sido ensayado con anterioridad al método X se supone que habrá una trasferencia total y el número de reglas nuevas del método X será igual a cero (p.e. salir es un método que siempre tiene número de reglas nuevas igual a cero dado que ha sido ensayado anteriormente).

• Si un método X presenta coincidencia parcial con otro método Y cuando ambos comienzan, el método practicado en segundo lugar descuenta el número de reglas en que se produce la coincidencia del total del método (esto al nivel 2 sólo ocurre con los procedimientos de Cambiar Comentario y de Abrir Archivo).