A Proposal to Elicit Usability Requirements within a Model-Driven Development Environment

Yeshica Isela Ormeño[1], Jose Ignacio Panach[2], Nelly Condori-Fernández[1,3], Óscar Pastor[1]

[1]Centro de Investigación en Métodos de Producción de Software ProS UniversitatPolitècnica de València Camino de Vera s/n, 46022 Valencia, Spain {yormeno, nelly, opastor}@pros.upv.es

[2]Escola Tècnica Superior d'Enginyeria Departamentd'Informàtica Universitat de València Av. de la Universidad, s/n 46100 Burjassot, Valencia, Spain joigpana@uv.es

[3]Faculty of Electrical Engineering, Mathematics and Computer Science University of Twente Information Systems Group 7500 AE Enschede, 217, Netherlands O.N.CondoriFernandez@utwente.nl

ABSTRACT

Nowadays there are sound Model-Driven Development (MDD) methods that deal with functional requirements, but in general, usability is not considered from the early stages of the development. Analysts that work with MDD implement usability features manually once the code has been generated. This manual implementation contradicts the MDD paradigm and it may involve much rework. This paper proposes a method to elicit usability requirements at early stages of the software development process such a way non-experts at usability can use it. The approach consists of organizing several interface design guidelines and usability guidelines in a tree structure. These guidelines are shown to the analyst through questions that she/he must ask to the end-user. Answers to these questions mark the path throughout the tree structure. At the end of the process, we gather all the answers of the end-user to obtain the set of usability requirements. If we represent usability requirements according to the conceptual models that compose the framework of a MDD method, these requirements can be the input for next steps of the software development process. The approach is validated with a laboratory demonstration.

Keywords*: Usability requirements, model-driven development, requirements elicitation process*

1.INTRODUCTION

Model-Driven Development (MDD) paradigm (Embley, Liddle, & Pastor, 2011) states that the analysts' entire effort should be focused on a conceptual model, and the system should be implemented by means of model to code transformations performed by a model compiler. A software production process is then seen as a set of conceptual models that are adequately transformed from requirements to code. A plethora of MDD methods and tools have been proposed, such as WebML (Ceri, Fraternali, & Bongio, 2000) or UWE (Koch, Knapp, Zhang, & Baumeister, 2008) among others.

There are two main dimensions to consider in MDD (Frankel, 2002): a “vertical” dimension and a “horizontal” dimension. In the vertical dimension there are at least three main layers that must be present in any MDD process:

1. A Requirements Modeling step, to produce a Requirements Model.
2. A Conceptual Model representation, where requirements are represented from the computer-oriented perspective.
3. The final Software Product (the Code).

The horizontal dimension focuses on the different expressiveness that must be present in the different conceptual perspectives of a MDD software process. Summarizing, these perspectives are:

- The data (static, system structure-oriented) perspective.
- The functional (dynamic, system behavior-oriented) perspective.
- The interaction (user interface-oriented) perspective.

While it can be argued that the two first perspectives (data and functionality) are largely explored by the different MDD approaches, it is surprising to realize that the interaction perspective is not at all so intensively explored. One could conclude that a Software Product is just the sum of a conceptual model where data and behavior are precisely specified, what is not exactly true, because the specification of the system interaction is an essential component of any software product. To confirm this situation, it is enough to consider the current modeling approaches that we find in practice. From the Data perspective, the question of what data models can be used to represent data has an immediate answer: ER and UML Class Diagrams are clearly among the most widely used and known. From the Functional perspective, since the appearance of the Data Flow Diagrams till the most modern UML diagrams designed to represent functionality, the offer is large. However, if the question is what models are specially used to represent System Interaction, the answer is not at all so immediate.

Extending a previous version presented at (Y. I. Ormeño, Panach, Condori-Fernandez, & Pastor, 2013), the goal of this paper is to explore the need of an interaction modeling, focusing on an essential software quality criteria that is mainly in the interaction scope: usability. Nowadays, in MDD, usability features are manually implemented once the code has been generated. According to Bass (Bass & John, 2003) and Folmer (Folmer & Bosch, 2004), these manual changes may involve changes in the system architecture, which can result in a lot of extra effort. Moreover, these manual implementations can produce a source code that contradicts the system’s characteristics expressed in the conceptual model.

In the previous work (Y. I. Ormeño et al., 2013) we defined how to elicit usability requirements according to existent usability guidelines. In this paper, we define how to include the usability requirements elicitation process in a MDD method. The main final goal of the paper is to define an approach to facilitate the usability requirements capture process for analysts who are not experts in usability engineering, and that want to include also the specification of usability requirements in a MDD-based approach.

The proposal to elicit usability requirements is based on the idea that first, an expert in usability defines a tree structure where design alternatives and usability guidelines are represented textually with questions and answers. Next, the analyst (non-expert in usability) can use this tree

structure indefinitely to ask end-users which alternative is the most suitable according to their requirements. Usability guidelines can help the end-user select an alternative throughout the tree structure. At the end of the process, we have a design for our system based on the end-user's requirements. If we represent the designs according to an existing conceptual model of a MDD method, those designs are the input for next development steps in the MDD process. The approach is validated with a laboratory demonstration with the participation of 4 subjects.

This paper is divided into the following sections: Section 2 presents the state of art of various approaches concerning both the modeling of interaction and the use of usability guidelines; Section 3 provides a general view of the approach to elicit usability requirements; Section 4 describes how to build the tree structure to represent all the design alternatives in an existent MDD method; Section 5 shows how to use the approach once the tree structure has been built; Section 6 reports an initial empirical validation of our approach. Finally, Section 7 describes the conclusions and future work.

2. RELATED WORK

The literature presents a lot of usability guidelines to support the design of user interfaces, but they may confuse the analyst if she/he is not an expert in usability. In general, the analyst may face the following problems (among others): it is not easy to understand how to apply the guideline; sometimes it is difficult to determine when a guideline has been broken; and, some guidelines are so ambiguous that they are difficult to apply to specific contexts. All these aspects require a huge effort on the part of the analyst that leads us to determine if the usability guidelines are still usable.

Cronholm's work (Cronholm, 2009) and Henninger's work (Henninger, 2000) describe possible solutions to some of these problems. Cronholm's work proposes meta guidelines as a solution to obtain more systematic and categorized guidelines. Design guidelines defined by Henninger include two types of guidelines: interface principles, or typed rules, and usability examples, also known as cases. These cases are examples of specific interfaces developed for organizations that contain a lot of knowledge about the needs and common practices of clients' work. Cysneiros's work (Cysneiros, Werneck, & Kushniruk, 2005) proposes a reusable catalogue to capture usability requirements. The method is based on i* framework and it uses personal experiences to obtain knowledge to achieve the objectives of usability.

The cited works aim to mellow the ambiguity of the usability guidelines, but they increase the complexity of use for non-experts in usability. All these solutions involve a lot of effort to understand all the guidelines and to choose the most suitable one for a specific context. For example, understanding the notation or the information arrangement in a guideline may involve some of the analyst's effort in order to use the guideline optimally. Furthermore, the comparison of guidelines shows great variability, which leads to creating specific usability guidelines for specific domains. Some authors aim to reduce developer's effort, such as Ferre (Ferre, Juristo, & Moreno, 2005), who defined a framework for usability practices integration. HCI techniques are characterized according to relevant criteria from a Software Engineering (SE) perspective and integrated into a framework organized according to development activities. Examples of methods to capture usability requirements are: a method for quantitative usability requirements applied in user interfaces to depict the true usability (Jokela, Koivumaa, Pirkola, Salminen, & Kantola,

2006); multimedia user interface designs that design attractive and usable multimedia systems (Sutcliffe, Kurniawan, & Jae-Eun, 2006); and, embedded Functionality Usability Features in model transformation technologies (Panach, España, Moreno, & Pastor, 2008). We can state that there are many proposals but none of them clearly and concisely addresses how to perform the usability requirements capture in early stages.

If we focus on approaches to elicit usability requirements according to the MDD paradigm, we realize that there are not previous works; in spite of MDD methods have usually a model to represent the interaction with the end-user. For example, WebRatio (Acerbis, Bongio, Brambilla, & Butti, 2007) includes a Presentation Model to express the layout and graphic appearance of pages, independently of the output device and of the rendition language. UWE (Koch et al., 2008) enables the definition of the front-end interface by means of a Hypertext Model. NDT (Escalona & Arag, 2008) has an abstract interface based on a set of prototypes to represent the interaction with the user. OO-Method (Pastor, 2007) has two models to represent the interaction: the Abstract Interaction Model (independently of platform) and the Concrete Interaction Model (platform-specific). All these MDD methods have some proposals to capture functional requirements but all of them lack of a process to capture usability requirements. This might result in unsatisfied end-users, which involves changes in conceptual models and in the generated code to solve problems related to interaction. This rework involves a lot of effort if analysts are not experts in usability. An early usability requirements elicitation guided by means of usability guidelines aims to prevent these problems from the first steps of the software development process.

This paper defines a process to organize the information stored in different usability guidelines based on a user-centred development (Hassenzahl, 2008). This way, analysts without a background in usability can work with the guidelines. Based on a review of the literature (Yeshica Isela Ormeño & Panach, 2013), we can say that very few papers that address how to perform the extraction process of usability requirements have been written (Henninger, 2000), (Cysneiros et al., 2005). Generally this task is done when the usability requirement capture has finished. Moreover, usability requirement capture has not been developed focusing on the MDD method. This paper aims to cover this gap, proposing a process to capture usability requirements such a way they can be transformed later into part of the conceptual model of the MDD method.

3. A PROPOSAL TO ELICIT USABILITY REQUIREMENTS

Based on ISO 9241-11 (ISO-9241_11, 1998) standard, usability requirements are requirements that affect effectiveness, efficiency and satisfaction of a user achieving his/her goals in a defined context of use. Our approach is based on existing usability guidelines and design guidelines, that are stored in a tree structure. The analyst navigates through this structure in order to capture the usability requirements by asking questions to end-users. The tree structure helps the analyst to identify the different design alternatives, and how these decisions will affect the system's usability. Figure 1 shows the elements used in our approach. Next, we describe each element:

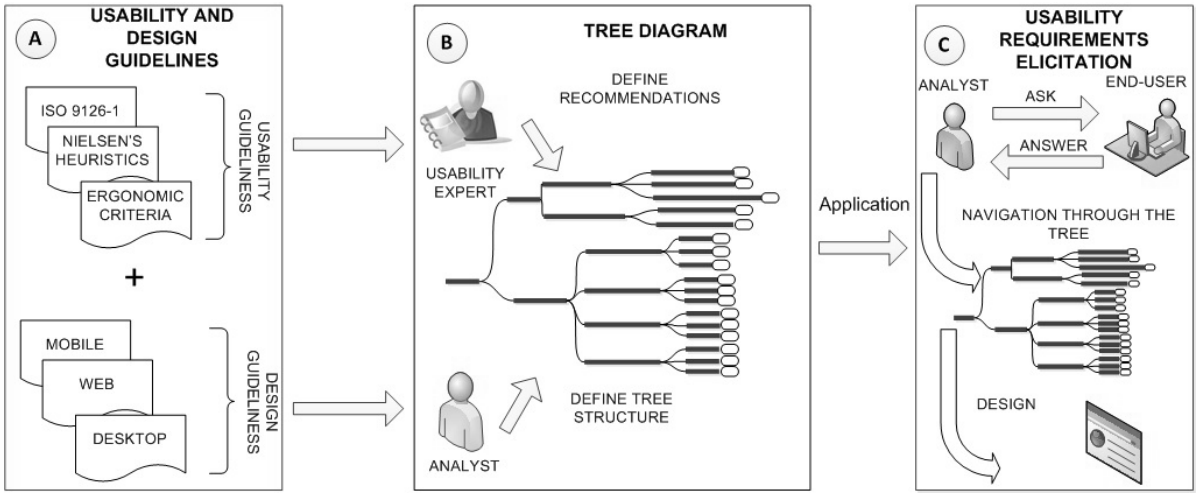

Figure 1. Schema of the proposal to capture usability requirements

3.1 Usability guidelines and interface design guidelines

Both usability guidelines and interface design guidelines have been created to guide the analyst to develop systems (Figure 1a). Usability guidelines recommend how to combine users, tasks and context to enhance the system usability. Interface design guidelines provide alternatives for designing systems. These guidelines have been built for different technologies and platforms that are represented by standards, principles, heuristics, styles, patterns, best practices, etc. Design and usability guidelines are related to each other since some design guidelines can improve or decrease the usability (depending on the combination of tasks, users and context). Working directly with both kinds of guidelines implies a huge effort as the variability and amplitude of these guidelines is very high. In order to reduce this effort, we propose storing all the relevant guidelines information in a tree structure, which is explained in more detail below.

3.2 Tree diagram

We propose using design and usability guidelines through a tree structure in order to minimize the cognitive effort to work with them (Figure 1b). A tree structure is defined as a connected graph with no cycles and a root (Johnsonbaugh, 1997). Figure 2 shows a general schema of the tree structure used in our approach, which is composed of four elements: question, answer, group of questions, and design. Next, we present these elements:

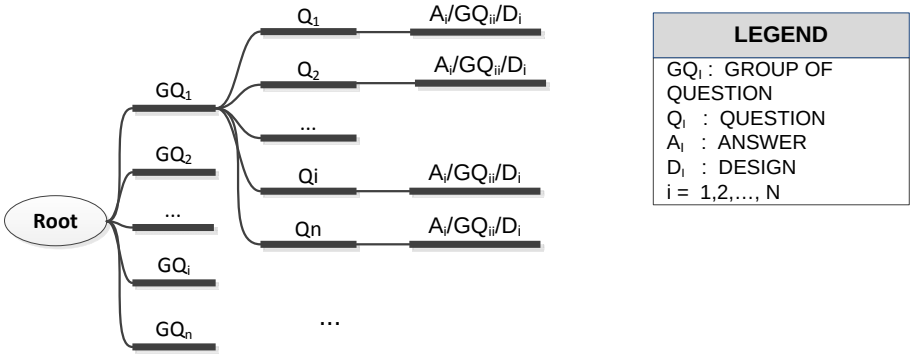

Figure 2. General representation of the tree structure (adapted from (Y. I. Ormeño et al., 2013))

1. Question(Qi): The design guidelines present diverse design alternatives for many UI (User Interface) components (e.g. menu). In order to ask the end-user which alternative she/he prefers, we have defined a question when alternatives to design appear. For example, when we are designing dialog elements for mobile, design guidelines (Nokia), (Android, 2012) specify that dialog elements provide a top-level window for short-term tasks and a brief interaction with the user. We can define a question to decide which is the UI component to represent a selectionable task, "*Which UI component is used to show selectable tasks?"*. In Figure 2, questions are represented by Qi.
2. Answer(Ai): These are the exclusive options for each question according to interface design guidelines. These options are presented to the analyst in such a way that she/he can choose which one best fits user's requirements. The analyst's decision is not only based on end-user criteria, but also on usability guidelines. This means that we must relate answers to usability guidelines depending on the type of user, task and context. When answers are shown to the analyst, we will show which answers are recommended by usability guidelines. For example, the answers to the question "*Which UI component is used to show selectable tasks?"* can be: RadioButtons, TextBoxes, CheckBoxes or Slider (Android, 2012), (Nokia). According to usability guidelines, a RadioButton is constructed for a persistent single-choice list (Android, 2012), where aspects such as "simplify navigation" and "minimize user input" are usability requirements (Cerejo, 2011). In Figure 2, answers are represented as Ai, Ai+1, … , An.
3. Group of Question (GQi): Some branches of the tree structure are not mutually exclusive (the end-user should be asked all of the questions). This type of branch is represented by a group of questions, which gathers several questions grouped by a design characteristic. For example, the question "*Which UI component is used to show selectable tasks?"* can be gathered with other questions that ask about Selection Dialogues, such as *"Where is the action button located?", "Where is the dialogue box located?",* and *"Where is the positive action on button located?"*. All these questions have in common that deal with how selection dialogs are displayed, and all of them are gathered in the same Group of questions. In the tree structure these are represented as GQi, in Figure 2.
4. Designs (Di): These are the interface designs reached through the alternatives that the analyst has been choosing. The analyst navigates through the tree structure asking the questions to the end-user, who selects the most suitable answer (usability guidelines can recommend some answers). When the analyst reaches a leaf in the tree, a design has been obtained. The final design of the whole system is the set of leaves in the tree that the analyst has reached. For example, a design can be a selection dialog with radio buttons, where each item shows an enumerated data (Nokia),(Android, 2012). At the tree structure these are represented as Di, in Figure 2.

The tree structure must be built by an analyst in collaboration with an expert in usability, who knows how to interpret and use usability guidelines. The expert in usability is responsible for defining the recommendations for each answer. In order to identify all the elements that compose the tree structure, we have defined a meta-model (Figure 3). The meta-model allows the replication of the tree structure in any context and the instantiation of as much instances as we need. Each instance can be used for different design and usability guidelines, resulting in different combinations of questions and answers.

Next, we describe the elements of the metamodel (classes). Class Design Guideline represents the interface design guidelines used in our tree structure. Questions that the end-user will be asked in order to discover which design alternative is most suitable are derived from these guidelines. Every question can be related to a group of questions, or to at least two Answers, since there is always more than one choice for each question. The class Group of Questions represents the set of questions we can define, and the class Answer specifies the exclusive alternatives for the question. Some of these answers can be recommended by one or several usability guidelines, recommendations, standards and best practices, represented as instances of the class Usability Guideline. According to the usability definition described in ISO-9241 (ISO-9241_11, 1998), some usability guidelines are specific for a context, task or user. This is represented through the classes Context, Task, and User respectively. Finally, class Design represents the designs that the analyst can get to at the leaves of the tree. Each instance of this class is a different interface design that we can reach through different answers.

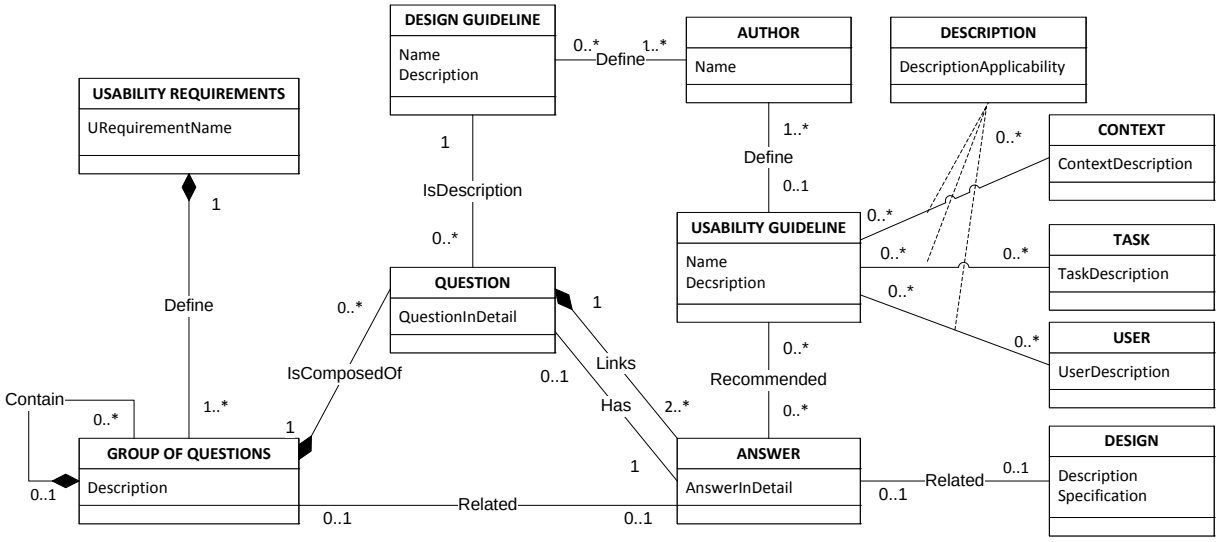

Figure 3. Meta-model of usability requirements capture (adapted from (Y. I. Ormeño et al., 2013))

3.3 Usability requirement elicitation

Once the tree structure has been finished, any analyst without explicit knowledge of usability can use it (Figure 1c). The usability requirement elicitation is the process to capture usability requirements using our approach. The navigation starts from the root of the tree while the analyst asks the questions to the end-users. The analyst asks the questions according to their sequence in the tree, from the root to the leaves. Questions are mutually exclusive, in other words, the analyst only navigates through the branch of the answer selected by the end-user. Questions that are gathered in the same group of questions are all asked. When the analyst reaches a branch with a group of questions, the flow continues with the first question in the group. Only when this flow has finished, the analyst can continue with the next question in the group. The possible navigation between two nodes of the tree structure can be: i) From a group of questions to a question, or to another group of questions (GQi→ Qi / GQi); ii) From a question to an answer

(Qi→ Ai); iii) From an answer to a question, to a group of questions or to a design (Ai→ Qi / GQi/ Di).

Note that if we work with several usability guidelines, they can contradict each other when they recommend an answer. For example, a widget with a ListBox (list of items) is recommended to improve Information Density (amount of information in the interface), since items are hidden inside the list. However, a RadioButton (◎) is recommended to improve Brevity (users' cognitive workload), since the items are displayed directly without the necessity of opening any list. This contradiction is not a problem in our approach, since usability guidelines are only recommendations. In case of contradiction, the analyst must tell the end-user which alternative is proposed by each usability guideline. The choice of the most suitable answer only depends on the user, who must choose according to his preferences. The analyst must explain to the user which usability recommendation satisfies each design alternative.

3.4 Including the Approach in a MDD Method

The link between the tree structure and a MDD method is performed through the leaves of the tree (the designs). Our approach consists in specifying the possible designs of the tree structure through a conceptual model of any existing MDD method. Most MDD methods have a specific model to represent end-user interaction (interaction model), that together with other models to represent persistency and behavior are the input for the code generation process. We propose using those interaction models to represent all the design possibilities expressed in the tree structure. Note that our proposal does not deal with how to work with interaction models or how to transform these interaction models into code. That depends exclusively on the MDD tool used as instantiation of our proposal. We focus on how to elicit usability requirements and how to include them in any of the existing MDD methods without modifying its existing conceptual model.

From all the MDD methods with an interaction model, we focus our illustrative example on OO-Method (Pastor, 2007). This choice is based on two characteristics: (1) OO-Method has an industrial tool named INTEGRANOVA (CARE, 2014) with a model compiler that can generate fully functional systems from a set of conceptual models without writing a single line of code. The generation is performed with ad-hoc transformation rules from models to code. All the models of the OO-Method framework are stored in a XML file that is the input for the code generation process. The XML file is read with a parser implemented in C++ that generates the code in C# or Java. (2) OO-Method has a model expressive enough to represent several design alternatives.

Next, we summarize both models of OO-Method to represent interaction: the Abstract Interaction Model (Molina, Meliá, & Pastor, 2002) and the Concrete Interaction Model (Aquino, 2008). The **Abstract Interaction Model** focuses on representing which are the elements that will be displayed for each interface. From a MDA perspective, this model is PIM since interfaces represented with this model are valid for any platform. These are the possible elements (named interaction patterns):

- Introduction: captures the relevant aspects of data to be entered by the end-user (including masks).
- Defined selection: enables the definition (by enumeration) of a set of valid values for an associated model element.

- Argument grouping: defines which input arguments can be grouped.
- Filter: defines a condition to display a list of elements.
- Order criterion: defines how a list can be ordered.
- Display set: determines the elements that compose a list with several fields.
- Actions: defines the set of available services.
- Navigations: determines the information set that can be accessed through a navigation between two interfaces.

The **Concrete Interaction Model** specifies how the elements that compose the interface will be displayed. From a MDA perspective, this model is PSM since interfaces represented with this model are for a specific platform. For example, in this model, the analyst decides the widget to display a Defined Selection (a list of enumerated values), which can be a ListBox or with a Radiobutton. The Concrete Interaction Model is defined through Transformation Templates, which specify the structure, layout and style of an interface according to preferences and requirements of end-users, and the different hardware and software computing platforms. A Transformation Template is composed of Parameters with associated values which parameterize the different design alternatives of the interfaces (Aquino, 2008). A part from interaction models, OO-Method is composed of an Object Model (which specifies the system structure in terms of classes of objects and their relations), a Functional Model (which specifies how events change object states) and a Dynamic Model (which represents the valid sequence of events for an object). A detailed description of all these models can be found in (Pastor, 2007).

Next, we apply the three elements of our approach (Figure 1) to OO-Method: (1) Usability and Design Guidelines; (2) Tree Diagram and (3) Usability Requirements Elicitation. This section deals with the two first elements, relegating the Usability Requirements Elicitation to next section. For the **first element** (Figure 1a) we use the design alternatives of the Abstract Interaction Model and the Concrete Interaction Model of OO-Method. As usability guidelines, we use ISO 9126-3 (ISO-9126, 2001) and the ergonomic criteria of Bastien and Scapin (Bastien, 1993). Both guidelines have been widely used in the software engineering community and in the human-computer interaction community.

The **second element** of our approach (Figure 1b) is the tree structure definition using design and usability guidelines previously chosen. From a MDA perspective, the tree structure is CIM, since it is independent of computation. According to (Y. I. Ormeño et al., 2013), the steps to build a tree structure are the following:

1. Identify design alternatives and define questions to ask the end-user which is the best design.
2. Express each design alternative as a possible answer for the questions defined previously.
3. Gather non-excluding design alternatives in groups of questions.
4. Define specific designs in the leaves of the tree.

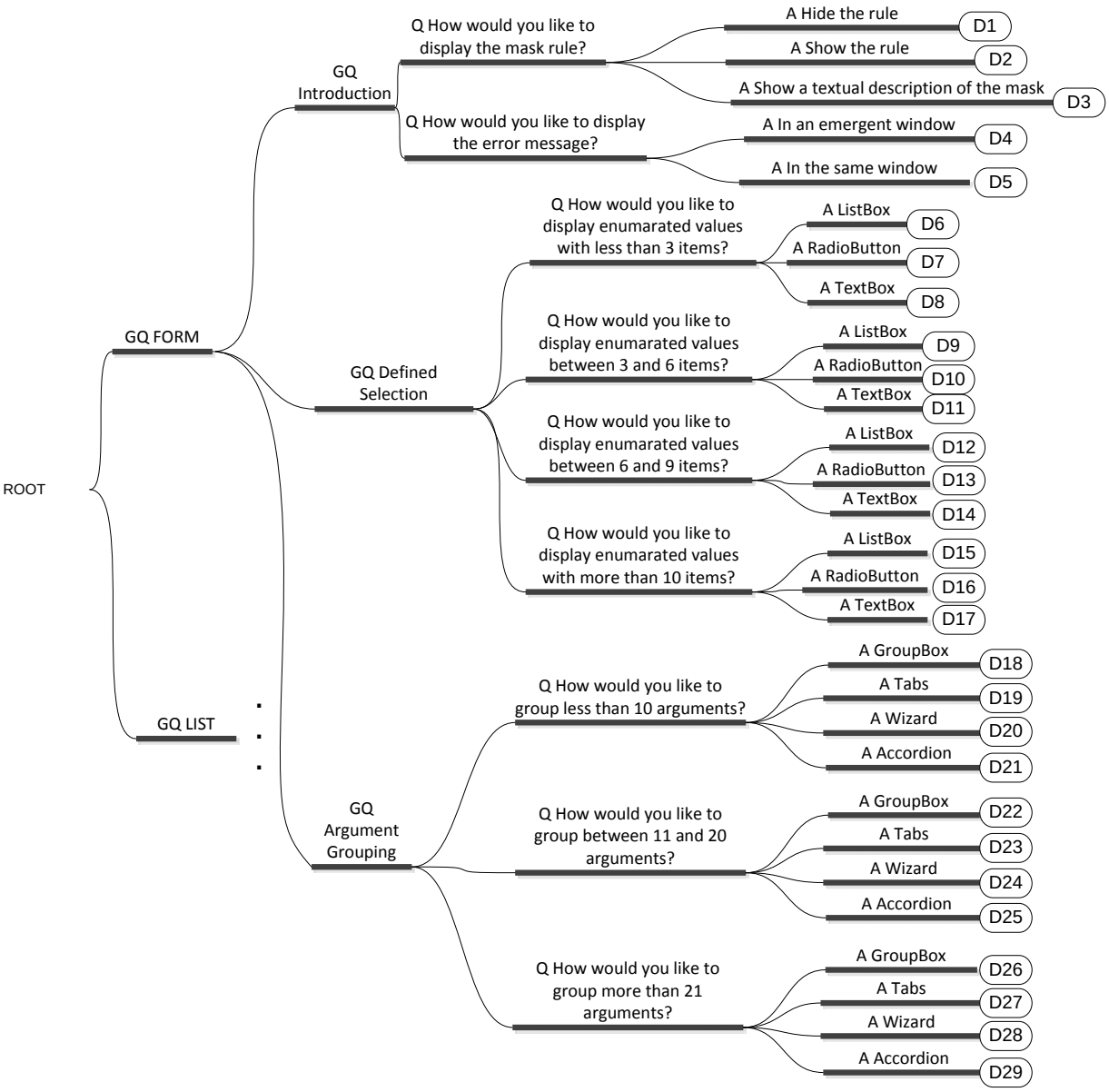

Figure 4. Tree structure with alternatives of OO-Method (1)

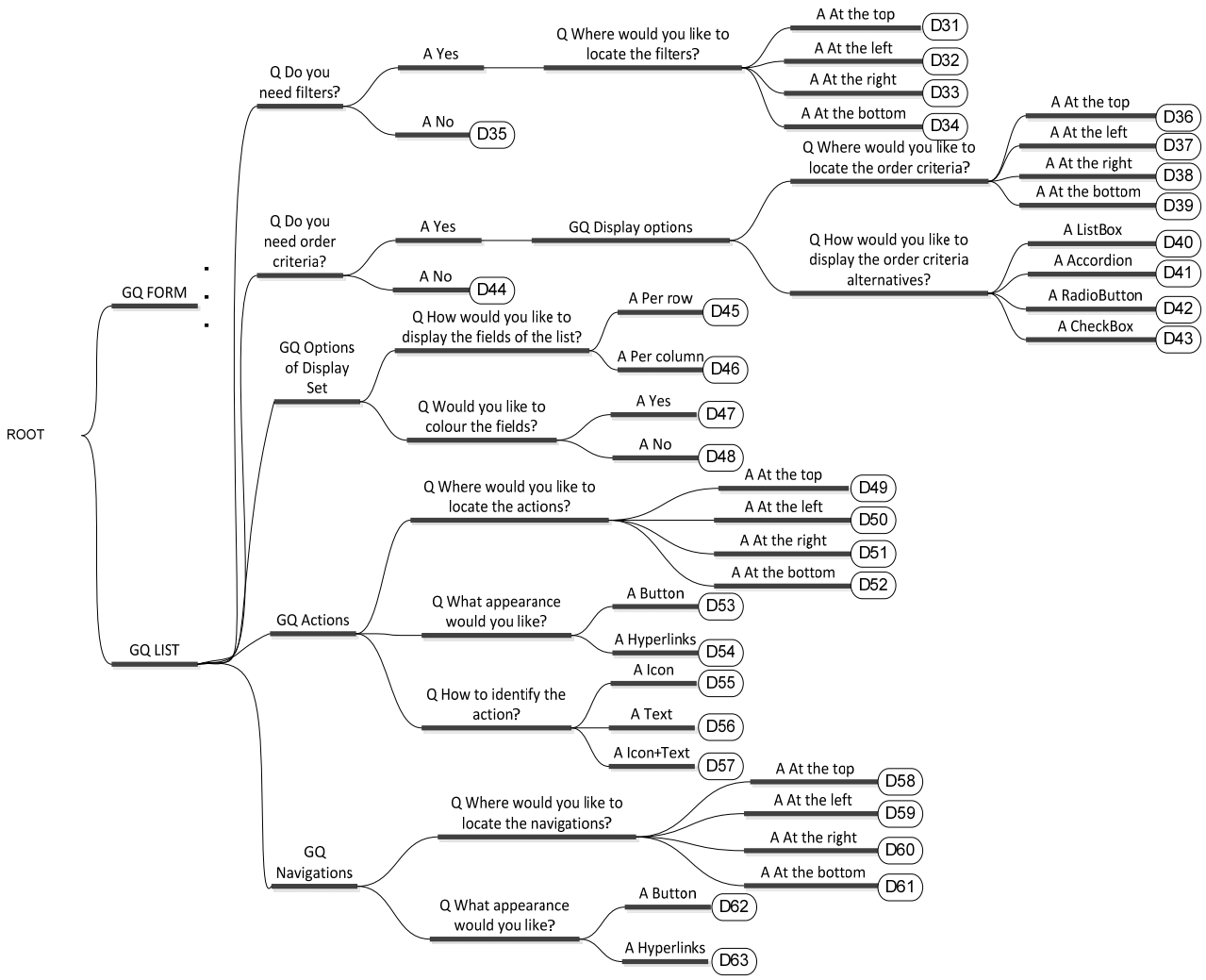

Figure 5. Tree structure with alternatives of OO-Method (2)

After applying all these steps to OO-Method, we have the tree structure displayed in Figure 4 and Figure 5. Each design is identified with the letter “D” and a number. Apart from identifying design alternatives, we have also identified the recommendations for the answers according to the metrics of ISO 9126-3 (ISO-9126, 2001) and the ergonomic criteria (Bastien, 1993). Next, we describe in detail the design alternatives identified in the Abstract Interaction Model of OO-Method and which ones are recommended according to usability guidelines. The tree structure has been performed by an analyst of OO-Method and an expert in usability. Each design alternative is represented in Figure 4 and Figure 5 as an answer:

- Introduction: the system can show the rule of a mask to prevent end-user from errors or hide it. Moreover, the error message displayed when inserted data does not fulfill the mask rule can be shown in a new emergent window or in the same window of the form. According to the ergonomic criterion Information Density, rules should not be shown, since they can overload the amount of information. However, criterion Error Protection (prevention of data entry errors) and metric Message Clarity (proportion of self-

explanatory messages) recommend showing the rules with a textual description to be understandable. Moreover, criterion Minimal Actions (workload regarding the number of actions) recommends showing the error message in the same window; while metric Interface Element Clarity (proportion of self-explanatory interface elements) recommends using a new emergent interface to show the error message.

- Defined Selection: the possible values can be inserted with a ListBox, a RadioButton or a TextBox (free text). According to criterion Minimal Action, enumerated values with less than 9 items should be displayed with RadioButtons, since all the possible values are shown directly (Panach, Condori-Fernández, Vos, Aquino, & Valverde, 2011). However, according to criterion Information Density, items should be displayed with a ListBox, such a way, the list of possible values is hidden until the end-user opens the list. Enumerated values with more than 9 items should be displayed with a ListBox according to the criteria Information Density and Legibility (lexical characteristics of information that facilitate the reading). In this case, a design with RadioButtons could increase the amount of information in the interface and a design with TextBoxes could not guide the user.
- Argument Grouping: arguments of a form can be grouped by a GroupBox (a group of elements in the same window), Accordion (a group of elements that can be hidden), Tabs (division of a form into different windows without relationship among them) or split into several interfaces through a Wizard (division of a form into different windows with a relationship among them). According to metric Functional Understandability (assessment that new users can understand the system) and criterion Guidance (availability of advising), a Wizard should be used when there are many arguments to perform an action. When there are not so many arguments, criterion Information Density recommends dividing the argument using Tabs or Accordion, since the end-users can show the arguments depending on their needs. When there are a few arguments, the design with a GroupBox is recommended by criterion Minimal Actions, since the arguments do not take up much space and they are shown directly.
- Filter: the first decision is to choose whether or not the system needs filters. Next, we must decide where displaying them. According to criterion Information Density, the use of a filter makes sense when there is a huge amount of information and the end-user needs some mechanisms to reduce it. However, when the amount of information is little, criterion Minimal Actions recommends not using a filter, such a way, end-users can list all the information directly. With regard to the position of the filter in the interface, top and left positions will consider the filter more important than the right and bottom positions. This recommendation provides from criterion Compatibility (match between users' characteristics and dialogues), that propose developing the system regarding end-users' perceptions and customs.
- Order Criterion: this pattern shares the same design alternatives as the filter, adding the possibility to choose how to display the different order criteria. According to criterion Legibility and metric Help Facility (proportion of functions described in the user documentation), order criteria should be used when there is much information in interfaces. This mechanism will help end-users identify quickly the required data. However, when the amount of information is little, criterion Minimal Actions

recommends not using Order Criteria, since the actions for ordering can spend more time than the benefit obtained with the order. With regard to the position of the order criteria, we can apply the same criterion used for Filter (Compatibility). How to display the order criteria alternatives will depend on the size of the screen. For wide screens, criterion Minimal Actions recommends displaying the order criteria with a RadioButton or a CheckBox. However, for narrow screens, criterion Information Density recommends hiding the order criterion until the end-user needs them. In this case, a design with a ListBox or Acordion is the most suitable.

- Display Set: the fields of the list can be displayed per rows or per columns. Moreover, we can colour the fields if we think that this will help to understand displayed data. According to criterion Compatibility, the fields of the Display Set should be compliant with the size of the screen in order to avoid scroll bars. Therefore, wide screens can show the different fields per column and narrow screens should show the fields per row. Moreover, criterion Legibility and metric Help Facility recommend using different colours per field to help end-users understand the information.
- Actions: there are different locations to display the actions; different widgets, such as buttons or hyperlinks; and different representations, such as icons, labels or a combination of icons and labels. According to criterion Compatibility, the recommendation for the position is the same as the recommendation for Filters. With regard to how to display the action in the screen, criterion Compatibility recommends using the widget most commonly used. Therefore, an appearance as Hyperlink is more suitable for Web applications and mobile systems, while an appearance as button is more suitable for desktop systems. Moreover, criterion Prompting (guide to make specific actions) and metric Function Understandability recommend identifying the actions such a way every user can recognize the action. Therefore, a textual label or an icon with a label is more suitable than only an icon. However, systems with a small screen should use icons according to criterion Information Density, since an icon will always take up less space than a textual description.
- Navigations: they share the same alternatives as actions. According to criterion Compatibility, the recommendation for the position is the same as the recommendation for Filters. Moreover, the recommendation for the appearance is the same as the recommendation for Actions according to criterion Compatibility.

The fourth step of our process consists in specifying the designs of the leaves through a conceptual model of the MDD method (Figure 6). This specification is the link between our proposal to elicit usability requirements and an existing MDD method. Each design of the tree structure can be represented in a conceptual model of the MDD method. Note that the process to specify the designs is done once only, when the tree structure is specified. How each design is specified depends exclusively on the used MDD method. As illustrative example, we describe how to specify the design to show a mask rule (D2 in Figure 4) and the design to display its error message in an emergent window (D4 in Figure 4). D2 and D4 must be specified both in Abstract and Concrete Models of OO-Method. This notation is just an example for the instantiation of our proposal to OO-Method:

- D2 is represented in the Abstract Model through the interaction pattern Mask, which is specified through the XML code:

<PIntroductionM id=”Mask_XX”>

<MsgError> “XXXX” </MsgError>

<PIntroduccionStringM Mask=” XXXX” /> </PIntroduccionStringM>

</PIntroduccionM>

D2 is represented in the Concrete Model through the template:

.MaskError=Mask_XX.MsgError

- D4 is represented with the same Abstract Model as D2, since both designs share the same interaction pattern: Mask.

  D4 is represented in the Concrete Model through the next template:

  .DisplayErrorMask= NewWindow

Note that models used to define the designs in the requirements elicitation step are initial interaction models composed of a first draft of Abstract and Concrete Models. By initial, we mean a model where specific details of the interface are not yet represented, just usability requirements. That is the reason why the previous examples of Abstract and Concrete Models do not specify an error message. In next development steps, the analyst must complete the interaction model and together with other models that represent persistency and behavior, they are the input for the model compiler. Finally, the model compiler interprets the characteristics expressed in the interaction models and generates the code that implements those characteristics. A detailed description about how to model interfaces with the Abstract Interaction Model (Molina et al., 2002), the Concrete Interaction Model (Aquino, 2008) and model to code transformations are out of scope of this paper since they do not concern the requirements elicitation step. Our contribution in this paper is only the process to elicit usability requirements (in grey background in Figure 6).

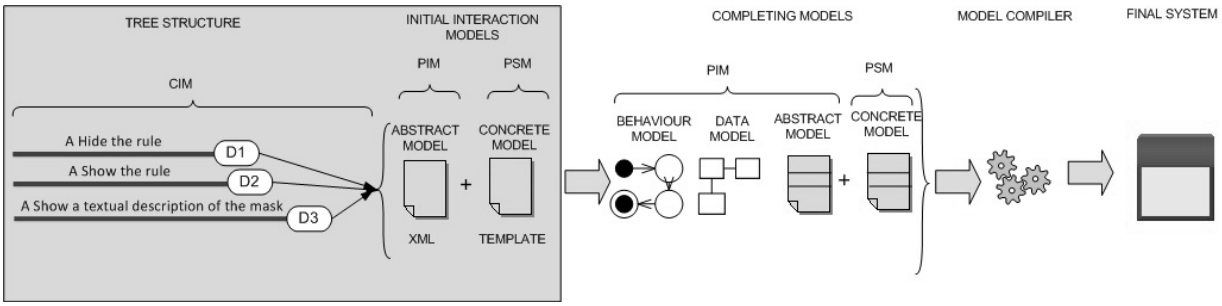

Figure 6. Overview of the process to include usability requirements in an MDD method

4. The Tree Structure in Use

This section describes the **third element** of our approach (Figure 1c): how to use the tree structure once it has been defined completely. As example, we use a system for car rental that must save information of all the cars that the car rental company has around the world; therefore,

the system needs to store much information. The system will follow a client-server architecture, such a way, the same server can connect with several clients in different platforms. In our example, we need to develop for two platforms: Web and mobile. The need of two platforms results in the development of two types of interfaces, in spite of the business logic is the same in both of them. In order to elicit the usability requirements for both systems, we must navigate two times through the tree structure of our approach.

First, we focus the example on eliciting usability requirements for the Web application. The process starts from the tree root to the leaves. When a question arises in the path, the analyst must ask the end-user that question. Apart from the question, the analyst must tell the end-user the possible answers to the question. If the answers are recommended by some usability guidelines, the analyst must specify which answers are recommended and why. Starting from the root (Figure 4 and Figure 5), we have a group of questions with two questions: *How would you like to display the mask rule?* and *How would you like to display the error message?* In this case, since the size of the screen is not a key issue, we can guess that the end-user chooses to show a textual description of the mask and to show the error message in a new window (A in Figure 7a). Once all the questions of a group of questions have been answered, the flow continues with the next question or group of questions with a pending answer. When a design (a leaf) arises in the path, the flow continues with the closest unresolved question.

In our example, the flow continues with the group of questions for Defined Selection. We guess that the end-user chooses as answers the recommendations for a Web application: using a RadioButton for items between 2 and 9 elements (B in Figure 7a), and using a ListBox for more than 9 items (C in Figure 7a). The next group of questions in the flow elicits requirements for Argument Grouping. According to the recommendations, the end-user selects a Wizard for more than 20 arguments, Tabs for a set between 11 and 20 arguments (D in figure 7a) and a Group Box for less than 10 arguments. Next, the flow continues with the questions regarding the Filters. Since there is much information to store in the system, the end-user selects to display the filters at the top of the interface (E in Figure 7b). This way, the first task end-users do within the interface is filling filters. Next, the flow continues with the questions regarding Order Criteria. Again, the amount of information recommends using order criteria. Since the size of the screen is not a problem, the end-user selects to display the order alternatives at the top of the interface using RadioButtons (which require less clicks than the use of a ListBox)(F in Figure 7b). Next, the flow continues with the questions regarding Display Sets. Since the screen for a Web application is wide, the recommendations suggest displaying the fields per column using different colours per field (G in Figure 7b). Next, the flow continues with the questions regarding Actions. According to the recommendations, the end-user selects to display the actions on the left with a hyperlink and to use a textual description (the size of the screen is not a problem) (H in Figure 7b). Finally, the flow continues with the questions regarding Navigations. The end-user selects to display the navigations at the bottom, since these actions will not be used very frequently (I in Figure 7b). Moreover, the visual appearance of navigations should be a hyperlink, since it is the most common widget for Web applications.

At the end of the process, we have a set of designs we have reached through the navigation of the tree structure. All these designs compose the set of usability requirements for the Web application. As example, we show the specification of designs D7, D10 and D13 used to display a RadioButton for lists between 2 and 9 items in INTEGRANOVA (B in Figure 7a). Note that all the designs are specified when the tree structure is defined. D7, D10 and D13 are represented in

the Abstract Model through the interaction pattern Defined Selection, which is specified through the XML code:

<PDefined_Selection id=”List_2-9”>

<Item1> “XXXX” </Item1>

<ItemN> “XXXX” </ItemN>

</PDefined_Selection>

This design is represented in the Concrete Model through the template:

.PDefined_Selection_id=”List_2-9”=RadioButton

This design is generic for every list of items between 2 and 9 elements. In next steps of the software development process, the analyst must complete this model for each interface that includes the pattern Defined Selection. In our example of Figure 7a, the Abstract Model will be completed with the following XML lines:

<PDefined_Selection id=”List_2-9” name=”Marital_Status”>

<Item1> Single </Item1>

<Item2> Married </Item2>

<Item3> Widowed </Item3>

</PDefined_Selection>

The Concrete Model does not need more details to specify how to display the list. The Abstract and Concrete Models are specified together with the other models of the OO-Method framework and finally we can obtain the final system. Figure 7 shows two examples of interfaces compliant with the requirements we have elicited for the Web application.

Figure 7.a,b Two examples of interfaces compliant with the requirements for a Web application

Figure 8.a,b Two examples of interfaces compliant with the requirements for a mobile application

Second, we use the tree structure again to elicit the usability requirements for the mobile system. In this case, the end-user would accept the recommendations for mobile applications, which claim to reduce as much information as possible in interfaces. In the group of questions Introduction, the end-user chooses to hide mask rules and to show error messages in a new emergent window (A in Figure 8a). Next, in the group of questions Defined Selection, the end-user selects to use ListBoxes in order to reduce the amount of information in interface (B in Figure 8a). Next, in the group of questions Argument Grouping, for a set of arguments between 2 and 20 items, the end-user chooses to use a design with Accordion (C in Figure 8a). Groups with more arguments should be displayed with a Wizard. Next, the end-user selects to display Filters at the top of the interface with an Accordion, since there is much information to display in little space (D in Figure 8b). Next, the end-user also selects Order Criteria at the top of the interface displayed with a ListBox, such a way they do not take up much space (E in Figure 8b). Display Sets are shown per row with colours, since mobile screens are very narrow (F in Figure 8b). Next, the end-user selects to show the Actions on the left of the interface, with a visual appearance of buttons and with a description based on icons (G in Figure 8b). Finally, for Navigations, the end-user selects to display them at the bottom of the interface using buttons, since this is the most frequently used representation for mobile systems (H in Figure 8b).

As example of designs specification, we show the specification of D6, D9, D12 and D15, used to display a ListBox for any group of items (B in Figure 8a). The Abstract Model for these designs is the same as the used for D7, D10 and D13. The Concrete Model is:

.PDefined_Selection_id=”List2-10”=ListBox

In next steps of the software development process, the analyst must complete the Abstract Model and the Concrete Model for each interface. For the example of list “Marital Status”, we can use the same Abstract Model as we defined for Defined Selection in Figure 7a. The Concrete Model does not need more changes. Figure 8 shows the same example of interface represented in Figure 7 but for a mobile system. Filters and Order Criteria have been hidden according to usability requirements.

5. INITIAL VALIDATION OF OUR APPROACH

Wieringa (Wieringa, 2010) classifies many different forms of validation that can be conducted with respect to a research proposal. This section describes a laboratory demonstration[1] that we have performed to validate the usability requirements elicitation process. We have used 4 subjects that are members of the PROS research center (http://www.pros.upv.es): 2 subjects play the role of analysts (persons that work usually with INTEGRANOVA) and other 2 subjects play the role of customers (persons without knowledge in INTEGRANOVA). We use two problems: Problem1 is a Web application to manage a car-rental system (like Figure 7) and Problem2 is a mobile application to manage a company of water supply. Table 1 shows the design used in the evaluation.

Treatments	**Without Tree**	**With Tree**
Problems	Problem1	Problem2
Subjects	Analyst1, Customer1	Analyst1, Customer1
	Analyst2, Customer2	Analyst2, Customer2

Table 1. *Evaluation design*

The experimental process consists in an interview between the analyst and the customer to elicit usability requirements of each problem with the target of developing both problems in INTEGRANOVA. Elicitation of Problem1 is performed without the tree structure and the elicitation of Problem2 is performed with the tree structure of Figure 4 and Figure 5 (design alternatives for INTEGRANOVA). Previously to the elicitation process, we explained how the tree structure works to the analyst. During the interview, the customer can change his requirements if the analyst offers him a better solution. Once the interview is over, we ask the analyst for interface sketches in paper. Next, the customer compares these sketches with his

[1] Technique used by the author on a realistic example in an artificial environment that shows that the technique could work in practice [a]

requirements. This way, we can confirm whether elicited usability requirements correspond to expected interfaces by the customer.

The **Factor** used in the experiment is the elicitation technique used for usability requirements. The factor has two levels: without our proposal and with our proposal. Each level is applied to each problem. **Response variables** are: time spent in the elicitation process (measured as minutes); design alternatives not asked to the customer and design alternatives that the customer changes after talking with the analyst (measured as number of design alternatives); analyst's satisfaction and customer's satisfaction (measured with a 5 point Likert scale). Table 2 shows the satisfaction questionnaires used.

Analyst's Satisfaction
I have no doubts about customer requirements
I would use the method to elicit requirements frequently
The method to elicit requirements is easy to use
The method to elicit requirements is useful
Customer's Satisfaction
The offered sketches satisfy your expectations
You would change your idea of system for the offered sketches
You think that the analyst has done a good work in the requirements elicitation process

Table 2. *Satisfaction questionnaires*

Results regarding **spent time** show that time spent using our approach is slightly higher (an average of 5 minutes more). Regarding **design alternatives not asked to the customer** without our approach, Analyst1 forgot asking 68% of design alternatives, and Analyst2 forgot 79%. Both analysts chose the most frequently used design alternatives without contrasting those decisions with the customer. Using our approach, both analysts asked 100% of design alternatives. Regarding **changes in interfaces during the interview,** Customer1 changed 5 features without our proposal and 6 features with our proposal. Customer2 changed 11 features without our proposal and 8 features with our proposal. Regarding **analyst's satisfaction,** both analysts are more self-confident with elicited requirements using our proposal, they would use our approach frequently and they classify our approach as useful and easy to use. Regarding **customer's satisfaction**, there are not differences between using our approach or not for the expected sketch and for the valuation of the analyst's work. Using our approach, both customers prefer the sketches of the analyst rather than their own ideas previous to the interview.

As conclusion, we state that even though this evaluation is a pilot experiment, results show an improvement in the elicitation process of usability requirements in a MDD method such as INTEGRANOVA: more matching between elicited requirements by the analyst and real needs of customers, and more satisfaction for analysts and customers. A disadvantage of our proposal is that it takes more time, since it requires asking the customers all the possible design alternatives.

Note that how to model the interaction and transformations have not been evaluated because they depend on the MDD tool used (INTEGRANOVA in this case).

6. CONCLUSIONS AND FURTHER WORK

This paper is a step forward to obtain holistic MDD methods, where all the system features, including usability, can be represented from the early steps till the code (vertical dimension). We propose a process to elicit usability requirements based on existent design alternatives and usability guidelines. The end-user must participate in the process, choosing the design alternative that better fits with her/his requirements. The approach is based on the construction of a tree structure that represents all the design alternatives. How to build the tree structure and how to use it, is explained in detail. Moreover, the approach has been validated with 4 subjects through a laboratory demonstration.

Note that the approach is valid for any MDD method but, as illustrative example, we have used OO-Method. This choice has led the design alternatives and the construction of the tree structure. The use of our approach in other MDD method, with models to represent the interaction different from the Abstract Model and the Concrete Model of OO-Method, involves building another tree structure. The size of the tree structure depends on the number of design alternatives; the more alternatives, the higher is the tree structure. One benefit of our proposal is that its use does not involve changing the existing MDD method. We do not propose any extension of existing interaction models or new transformation rules. We propose using existing interaction models to represent designs of our tree structure, and those models will be the input for existing transformation rules in next steps of the development process (if the existing MDD method supports these transformations).

In our example, we have used two usability guidelines: ISO 9126-3 and the ergonomic criteria. In Human-Computer Interaction and in Software Engineering communities there are many other guidelines. Our approach accepts as many guidelines as the analyst would like to consider. A contradiction between two guidelines does not mean a problem, since the end-user decides the most suitable design alternative. However, it is important to mention that too many recommendations for the possible designs can confuse end-users.

Our approach focuses on eliciting usability requirements. As outcome of our elicitation process we get some incomplete conceptual models. In next development steps, the analyst must enhance these models with primitives that represent the functionality and the visual appearance of the system in order to get a fully functional system. How the usability requirements will be expressed in the next steps of the software process will depend exclusively on the MDD method.

As future work, we plan to develop a tool to support the construction and use of any tree structure. Even with a few design alternatives and a few usability guidelines, the size of the tree structure is considerable. Moreover, we also plan to apply our proposal to a real case study in industry with more subjects than the ones used in this paper.

ACKNOWLEDGEMENTS

This work has been developed with the support of MICINN (PROS-Req TIN2010-19130-C02-02), UV (UV-INV-PRECOMP13-115032), GVA (ORCA PROMETEO/2009/015), and co-financed with ERDF. We also acknowledge the support of the Intra European Marie Curie Fellowship Grant 50911302 PIEF-2010. We also thank Sergio España, Francisco Valverde, Marcela Ruiz and María Jose Villanueva for their participation in the experimental validation.

REFERENCES

Acerbis, R., Bongio, A., Brambilla, M., & Butti, S. (2007). WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications. *LNCS, 4607*, 501-505.

Android, D. (2014). User Interface Guidelines, from http://developer.android.com/guide/practices/ui_guidelines/index.html

Aquino, N., Vanderdonckt, J., Valverde, F., Pastor, O. (2008). *Using Profiles to Support Model Transformations in the Model-Driven Development of User Interfaces.* Paper presented at the Proc. of 7th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2008, Albacete, Spain.

Bass, L., & John, B. (2003). Linking Usability to Software Architecture Patterns through General Scenarios. *Journal of Systems and Software, 66*(3), 187-197.

Bastien, J. M., Scapin, D. (1993). Ergonomic Criteria for the Evaluation of Human-Computer Interfaces. *Rapport technique de l'INRIA*, 79.

CARE. (2014). CARE Technologies, from https://www.care-t.com.

Cerejo, L., A. (2011). User-Centered Approach To Web Design For Mobile Devices. Retrieved 11 october 2012, from http://mobile.smashingmagazine.com/2011/05/02/a-user-centered-approach-to-mobile-design/

Ceri, S., Fraternali, P., & Bongio, A. (2000). Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. *Computer Networks, 33*(1), 137-157.

Cronholm, S. (2009). *The Usability of Usability Guidelines: A Proposal for Meta-guidelines.* Paper presented at the 2lth Australasian Conference on Computer-Human Interaction, Melbourne, Australia.

Cysneiros, L. M., Werneck, V. M., & Kushniruk, A. (2005, Aug 29 - Sept 2, 2005). *Reusable Knowledge for Satisficing Usability Requirements.* Paper presented at the 13th IEEE International Conference on Requirement Engineering, Washington, DC, USA.

Embley, D. W., Liddle, S. W., & Pastor, O. (2011). Conceptual-Model Programming: A Manifesto. In D. W. Embley & B. Thalheim (Eds.), *Handbook of Conceptual Modeling* (pp. 3-16): Springer Berlin Heidelberg.

Escalona, M. J., & Arag, G. (2008). NDT. A Model-Driven Approach for Web Requirements. *IEEE Trans. Softw. Eng.*, *34*(3), 377-390.

Ferre, X., Juristo, N., & Moreno, A. M. (2005). *Framework for integrating usability practices into the software process*. Paper presented at the Proceedings of the 6th international conference on Product Focused Software Process Improvement.

Folmer, E., & Bosch, J. (2004). Architecting for Usability: A Survey. *Journal of Systems and Software, 70,* 61-78.

Frankel, D. (2002). *Model Driven Architecture: Applying MDA to Enterprise Computing*: John Wiley & Sons, Inc.

Hassenzahl, M. (2008). The interplay of beauty, goodness, and usability in interactive products. *Hum.-Comput. Interact.*, *19*(4), 319-349.

Henninger, S. (2000). A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design. *Interacting with Computers, 12*(3), 225-243.

ISO-9126. (2001). Software Engineering - Product Quality - Part 1: Quality Model.

ISO-9241_11. (1998). Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) - Part 11: Guidance on Usability.

Johnsonbaugh, R. (1997). *Discrete Mathematics* (Fourth ed.). New Jersey: Prentice Hall Intemational.

Jokela, T., Koivumaa, J., Pirkola, J., Salminen, P., & Kantola, N. (2006). Methods for Quantitative Usability Requirements: A Case Study on the Development of the User Interface of a Mobile Phone. *Personal Ubiquitous Comput.*, *10*(6), 345-355.

Koch, N., Knapp, A., Zhang, G., & Baumeister, H. (2008). UML-Based Web Engineering, an Approach Based on Standards *In Web Engineering, Modelling and Implementing Web Applications* (pp. 157-191): Springer.

Molina, P. J., Meliá, S., & Pastor, Ó. (2002). *JUST-UI: A User Interface Specification Model.* Paper presented at the Proceedings of Computer Aided Design of User Interfaces, CADUI'2002, Valenciennes, Francia.

Nokia. (2014). Symbian Design Guidelines - Dialogs. from http://www.developer.nokia.com/Resources/Library/Symbian_Design_Guidelines/

Ormeño, Y. I., & Panach, J. I. (2013). *Mapping study about usability requirements elicitation*. Paper presented at the Proceedings of the 25th international conference on Advanced Information Systems Engineering.

Ormeño, Y. I., Panach, J. I., Condori-Fernandez, N., & Pastor, O. (2013, 29-31 May 2013). *Towards a proposal to capture usability requirements through guidelines.* Paper

presented at the IEEE Seventh International Conference on Research Challenges in Information Science (RCIS'2013)

Panach, J. I., Condori-Fernández, N., Vos, T., Aquino, N., & Valverde, F. (2011). Early Usability Measurement In Model-Driven Development: Definition and Empirical Evaluation. *International Journal of Software Engineering & Knowledge Engineering (IJSEKE)*.

Panach, J. I., España, S., Moreno, A., & Pastor, O. (2008). *Dealing with Usability in Model Transformation Technologies.* Paper presented at the ER 2008, Barcelona.

Pastor, O., Molina, J. (2007). *Model-Driven Architecture in Practice*. Valencia: Springer.

Sutcliffe, A. G., Kurniawan, S., & Jae-Eun, S. (2006). A Method and Advisor Tool for Multimedia User Interface Design. *Int. J. Hum.-Comput. Stud., 64*(4), 375-392.

Wieringa, R. (2010). *Design science methodology: principles and practice*. Paper presented at the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering.