Towards a Proposal to Capture Usability Requirements Through Guidelines

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

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

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

***Abstract*—The Model-Driven Development (MDD) paradigm states that analysts can build a conceptual model that represents the system abstractly. This conceptual model is the input for a set of transformation rules that can generate the code that implements the system automatically. Nowadays, there are sound MDD methods that deal with functional requirements, but, in general, usability is not taken into consideration from the early stages of the development. Analysts who work with MDD implement usability features manually once the code has been generated. This manual implementation contradicts the MDD paradigm, and it can affect the system architecture, involving a lot of reworking. This paper proposes a method to capture usability requirements at the early stages of the software development process in such a way that non-experts in 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 the end-users. Answers to these questions mark the path through the tree structure. At the end of the process, if we gather all the end-user's answers, we have the usability requirements. Then, by means of model to model transformations, we could transform usability requirements into a conceptual model of any existing MDD method.**

Keywords—Usability requirements, model-driven development, requirements capture process

I. INTRODUCTION

The Software Engineering (SE) community has been working for several years on the Model-Driven Development (MDD) paradigm [1], which 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. In MDD, a conceptual model is used to represent a system, independent of the platform and technology. This conceptual model is the input for a model compiler which includes transformation rules to generate the code according to the target platform.

Even though existing MDD methods (e.g. WebML [2] or UWE [3]) are very powerful for building conceptual models, they do not have a process to capture usability requirements. In general, usability features are manually implemented once the code has been generated. This manual implementation contradicts the MDD paradigm, which proposes focusing the analyst's entire effort on building a holistic conceptual model. According to Bass [4] and Folmer [5], 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.

So, why are usability requirements not captured in the early software development stages together with functional requirements? One reason for this is that usability is strongly related to human behavior (software psychology [6]) and, unfortunately, analysts who capture system requirements are not experts in this field. In order to facilitate the software development process, the Human Computer Interaction (HCI) community has defined usability guidelines for non-experts in usability. For example, Shneiderman [7], and Nielsen's [8] usability design guidelines are widely accepted and used as tools to measure usability. However, these guidelines are usually described in such an abstract way that they are difficult to apply (directly) in software development. Moreover, the evolution and presence of new technologies and communication devices encourages the development of usability guidelines oriented to different platforms (contexts) such as: the Web, development tools, phones, tablets and media devices [9]. According to Nielsen [10], there are around 2394 guidelines. The Web is the software platform with the most guidelines. It contains 874 user-experience design guidelines, 144 guidelines for commercial businesses, 103 for corporate sites and 614 usability design guidelines on the intranet. This huge number of guidelines hinders the analyst when he/she is searching for the most suitable guideline for a specific system.

Thus, the main contribution of this work is to define an approach to facilitate the usability requirements capture process for analysts who are not experts in usability engineering. This approach can be included in an MDD method in such a way that these requirements generate part of the conceptual model of the MDD method. This is in accordance with the MDD paradigm, which states that models used in the early stages of the software development process can be transformed into models for the next stages. The approach is based on textual

questions, and design alternatives for each question that end-users must be asked relevant questions, and design alternatives, are extracted from interface design guidelines and they are represented in a tree structure. End-users must choose which alternative is the most suitable according to their requirements (or constraints). 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. This design can be embedded in a conceptual model of an existing MDD method through transformation rules.

This paper is divided into the following sections: Section 2 presents the state of art of various approaches made by other authors concerning the use of usability guidelines; section 3 describes the concepts that are involved in the usability requirement capture approach; section 4 explains the proposed scheme to capture usability requirements viewed from both the analyst's and the expert's side; section 5 presents a proof of concept based on an example, and finally, Section 6 describes the conclusions and future work.

II. 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 [11] and Henninger's work [12] describe possible solutions to some of these problems. Cronholm's work proposes meta guidelines as a solution to obtain more systematic and categorized guidelines. These meta guidelines consist of a set of principles whose objective is to improve the usability of the 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.

Furthermore, Cysneiros's work [13] 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. His work shows how usability can be modeled through different views with different alternatives. Bevan [14] makes a comparison between three guidelines: HHS for a Web site, JISC for Web services, and ISO 9241-151, which includes principles and specific solutions (conceptual models, task structure, and navigational structures). Bevan highlights differences and similarities between these three guidelines. He states that a perfect set of guidelines does not exist, since the necessities of different audiences are not homogeneous.

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 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.

Usability guidelines for the Web and for WAP mobile phone applications are widely used. Pei [15] states that web design should be focused on the user Web site to improve usability. The design of a usable web is made up of the following three elements: user research, web design, and usability evaluation. On the other hand, the usability of mobile phone applications is increasing, although it is lower than Web Sites accessed by computer [16]. Sabine's work [17] proposes usability guidelines to design applications based on WAP. This author compares two versions of a travel management Web Site, one which includes usability guidelines of design and the other which does not. The results show that user-experience of the Web site which uses usability guidelines is higher than mobile phone or Smartphone applications with standard features.

The literature provides a wide range of usability guidelines for web sites, web applications, desktop applications, mobile phones and others [10]. Some examples of usability guidelines are: development tools (AJAX, RIA), User Interface (Apple Mac OSX, iPad user experience) platform (Window XP, Vista User Experience Interaction) Interface Software Mobile (Android, Nokia top 10, WebOS) among others [9]. Moreover, these existing guidelines are continuously in state of change and development especially for mobile phone Internet services looking to improve usability.

Some examples of methods used to capture usability requirements are: a method for quantitative usability requirements applied in user interfaces to depict the true usability [18]; multimedia user interface designs that design attractive and usable multimedia systems [19]; and, embedded Functionality Usability Features in model transformation technologies [20]. We can state that there are many proposals but none of them clearly and concisely addresses how to perform the extraction process of usability requirements in the early stages.

This paper proposes a method to organize the information stored in different usability guidelines. This way, analysts without a background in usability can work with the guidelines. Based on a review of the literature, we can say that for the MDD paradigm very few papers have been written that address how to perform the extraction process of usability requirements. Generally this task is done when the usability requirement capture has been done. 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.

III. PROPOSAL TO CAPTURE USABILITY REQUIREMENTS

This section describes our approach to capture usability requirements within the MDD paradigm. Based on the ISO 9241-11 [21] standard, the usability requirements are the 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 the end-users questions. 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:

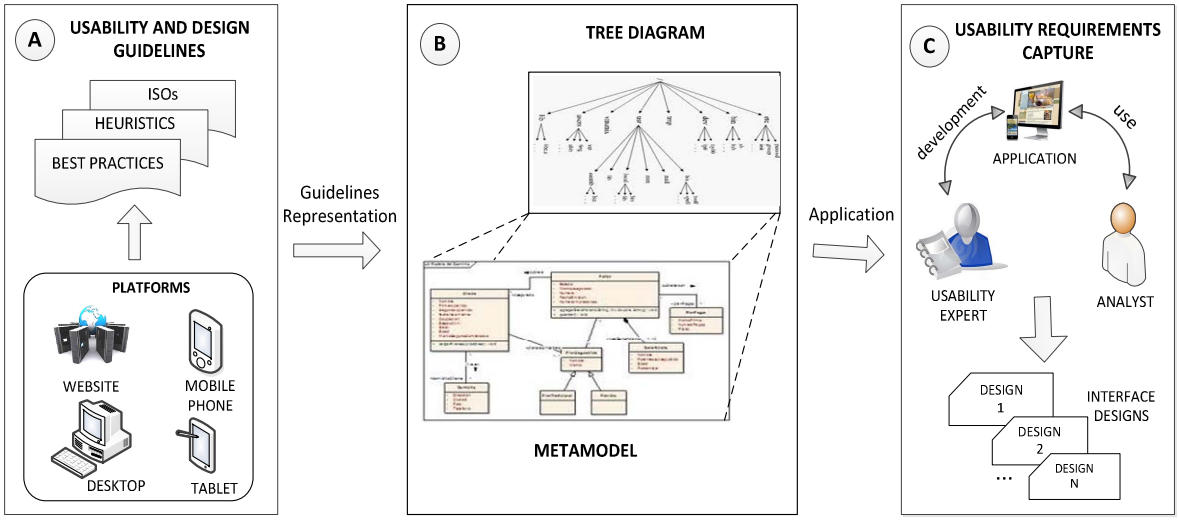


Fig. 1. Schema of the proposal to capture usability requirements.

A. *Usability guidelines and interface design guidelines*

Both usability guidelines and interface design guidelines have been created to guide the analyst to develop systems. Usability guidelines recommend how to combine users, tasks, and context to enhance the system usability [21]. Interface design guidelines provide alternatives and recommendations for design systems [22]. These guidelines have been built for different technologies and platforms which are represented by standards, principles, heuristics, styles, patterns, best practices, etc. Both types of 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 [23], [24], [21], 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 guideline information in a tree structure, which is explained in more detail below.

B. *Tree Diagram*

In this context, we propose using these guidelines by means of a tree structure in order to minimize the cognitive effort to work with both types of guidelines. A tree structure is defined as a connected graph with no cycles and a root [25],[26]. 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 designs. In the next part, we will present these elements:

1) Question(Q_i): 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 [27], [24] 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?*. This question could enable the user to complete a specific task. In Figure 2, questions are represented by Q_i.

2) Answer(A_i): 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 the user's requirements. The analyst's decision is not only based on end-user criteria, but also on usability guidelines. This means that we have related answers with usability guidelines depending on the type of user, type of task, and type of context. When the 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: radio buttons, text field, checkboxes, slider [24],[27]. Mobile design guidelines [28] advise using a UI component dialogue to show tasks as

information that require users to take an action before they can proceed. The usability recommendations are identified when answers have been defined. For example a radio button is constructed for a persistent single-choice list [24], where aspects such as *"simplify navigation"* and *"minimize user input"* are usability requirements [28]. In Figure 2, answers are represented as A_i, A_{i+1}, … , A_n.

3) Group of Question (GQ_i). 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 also in common that deal with how the selection dialogs are displayed, and all of them are gathered in the same Group of questions . In the tree structure these are represented as GQ_i, in Figure 2.

4) Designs (D_i): 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 [27],[24]. At the tree structure these are represented as D_i, in Figure 2.

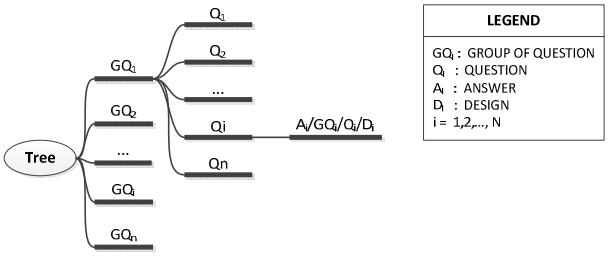


Fig. 2. General representation of the tree structure of a figure caption.

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, can the analyst 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 ($GQ_i \rightarrow Q_i$ / GQ_i); ii) From a question to an answer ($Q_i \rightarrow A_i$); iii) From an answer to a question, to a group of questions or to a design ($A_i \rightarrow Q_i$ / GQ_i / D_i).

Note that if we work with several usability guidelines, they can contradict each other when they recommend an answer. This contradiction is not a problem in our approach, since usability guidelines are only recommendations. The choice of the most suitable answer only depends on the analyst and on the user's requirements.

One advantage of our approach is that designs reached throughout the navigation in the tree can be transformed into a conceptual model of a MDD method. For this aim, each design of the tree must have a transformation rule to generate part of the conceptual model of the target MDD method, as Figure 3 shows. In order to facilitate these transformations, we recommend using UsiXML (USer Interface eXtensible Markup Language) [29] as the language to specify the designs. UsiXML is an XML-based markup language for defining user interfaces which is widely used in the academy. The main advantage of using UsiXML is that a framework has already been defined to support interface modeling, and there are also transformations from UsiXML to some MDD methods, which facilitates the transformation work.

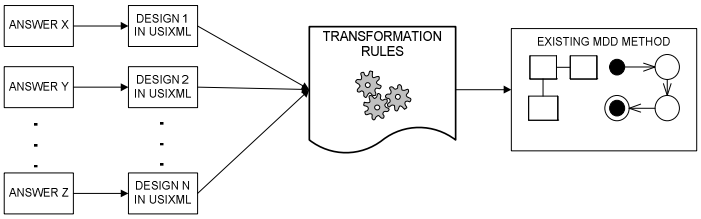


Fig. 3. General process to generate a conceptual model from the designs.

In order to formalize all the elements that compose the tree structure, we have defined a meta-model (Figure 4). Below, we describe its 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. 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 [21], some usability guidelines are specific for a context, task or user [30],[31]. This is represented through the classes Context, Task, and User respectively. The class Context describes the context where the guideline is recommended, the class Task describes the type of task for which the guideline is recommended, and class User describes the type of user for which the task is recommended. Context, Task and User are related to class Description, to describe how they enhance the system's usability. 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 which we can reach through different answers.

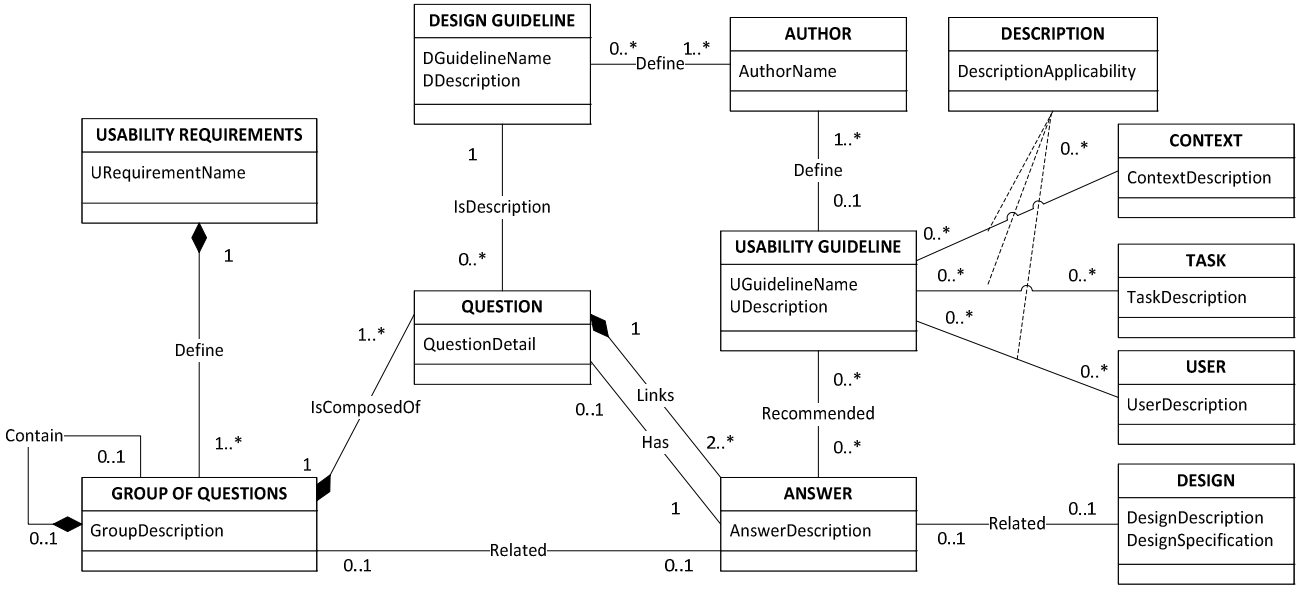


Fig. 4. Meta-model of usability requirements capture.

C. *Usability requirement capture*

The usability requirement capture is the process to capture usability requirements using our approach. The next section explains how to build the tree structure, and how to use it in the requirement capture process

IV. PROCESS TO CAPTURE USABILITY REQUIREMENTS IN MDD

This section describes the process to build an instance of the meta-model shown in Figure 4. This instance will be used later to capture usability requirements. Three stakeholders participate in this process: an expert in usability, an analyst and the end-user. In the next section we will explain how the stakeholders participate in both activities: the construction of the tree structure and requirement capture.

A. *Phase of construction*

This phase is performed by the usability expert and the analyst. First, the usability expert builds the tree structure using interface design guidelines and usability guidelines.

Second, the analyst specifies the transformation rules to transform the designs into a conceptual model of a MDD method. Figure 5 summarizes all the steps that make up this phase. Below we detail all of them.

SE1)Analysing the usability guidelines and interface design guidelines: The usability expert looks for existing interface design guidelines and usability guidelines that can be applied to build the tree structure. In the literature there are many guidelines, the expert must choose on those guidelines focused on the type of systems we aim to build using the tree structure. Then, an analysis of these guidelines is required to identify the relevant aspects for designing usable systems. It is important to point out that this identification of relevant aspects depends on the experience level of the "usability expert" to appropriately construct the tree. The identification of these relevant aspects depends on the experience of the usability expert.

SE2)Defining the question: Using interface design guidelines, the usability expert defines the questions. When there is a set of possible alternatives for a design, the expert must define a question in order to ask the user which is the most suitable alternative.

SE3) Defining the answer: Each alternative to a question is expressed as a possible answer for that question. According to the tree structure, after specifying an answer the usability expert has several possibilities: (1) To define another more specific question (if we need more information to determine the final design); (2) To define a final design (if we have reached a leaf in the tree because there are no more alternatives); (3) To define a group of questions (if the answer leads to more than one related questions).

SE4)Recommending usability guidelines: Usability guidelines may recommend some answers. In this step, the usability expert defines which answers are recommended by which usability guideline. Recommendations can be given with respect to any of the elements: context, task, or user. The relationship between answers and usability guidelines is not mandatory, but the more guides we provide to the end-user to choose the answer the more possibilities to build a usable system we have.

SE5) Defining the group of question: The usability expert defines the groups according to the topic of the questions. Note that the end-user will be asked every question included in a group.

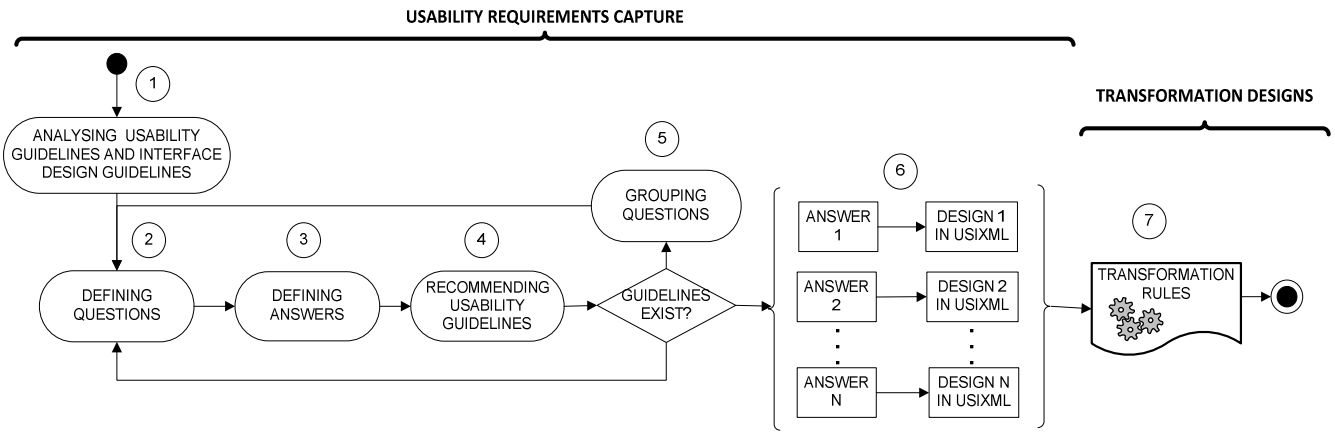


Fig. 5. Process to build the tree structure to capture usability requirements.

SE6)Obtaining interface designs: When the usability expert identifies that there are no more alternatives to specify a design, she/he can define this design formally. Each design (leaf) of the tree structure must be completely different to other designs, since the path used to reach the design will be exclusive. We propose defining these designs using the UsiXML [29] language. This definition must be performed by the analyst, since the usability expert does not work with conceptual models usually, and this topic is out of the scope of his / her expertise.

SE7) Transformation rules definitions: Once the designs have been defined, the analyst must specify transformation rules to transform these designs into primitives of the conceptual model of a MDD method. The transformations aim to include all the usability requirements in the software development process. Since we propose specifying the designs with UsiXML some of these transformations already exist [32].

B. Phase of use.

This phase explains how the analyst uses the tree structure to capture usability requirements. The process starts from the tree root to the leaves. When a question arises in the path, the analyst must ask the end-user the 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. Note that more than one answer can be recommended, and some usability guidelines can contradict each other. This is not a problem, since the end-user must choose the answer that best fits the requirements, independent of the recommendations. When the end-user chooses an answer, the flow continues through the branches of that answer, while the branches of the other rejected answers will not be crossed.

When a group of questions arises in the path, the analyst must ask the end-user every question in this group to based on the order they were created. Once the analyst asks the first question in the group, the flow continues with the branch of that question. When this branch has been completely gone through, the flow continues with the second question in the group. This process is repeated for every question in the group.

When a design arises in the path, the flow continues with the closest unresolved question. At the end of the process, we have a set of designs we have reached through the navigation. These designs are then transformed into primitives of a conceptual model of a MDD method according to the transformation rules previously defined. Note that rules are defined once, but they can be used indefinitely for the same tree structure and the same MDD method.

V. A LABORATORY DEMONSTRATION

In order to illustrate the usability requirements capture process, we show an example to design a menu for a mobile phone application. Next, we exemplify our proposal for capturing usability requirements:

A. Phase of construction

SE1) Analizing the usability guidelines and interface design guidelines: As there are many interface design guidelines specific for mobile devices, our analysis focus only on Android[24], iOs [23], and Symbian [27] guidelines, since they provide specific descriptions to design menus and are widely used. With respect to usability guidelines, we used Nielsen's heuristics [33] since it is widely known and used by user-interface designers to develop usable systems.

From the interface design guidelines [24], we identified the most relevant aspects that should be considered in order to capture usability requirements. In our example, we focus on the "display mode" as a relevant aspect, since there are different ways to display menu options.

SE2) Defining the questions: We define define the questions to ask concerning how to display the menu options in a system. According to interface design guidelines [24], we have identified the following questions: Q1. *How can the menu options be displayed?*; Q2. *What is the layout type to display nest views?*; Q3. *How is the contextual action item displayed?*

Q1 has been extracted from Symbian [27] guidelines, which state that menu options are "an efficient way to allow users to perform actions". Therefore, the definition of the menu display is essential to allow users to trigger actions. Q2 has

been extracted from the Android guideline [24], which proposes defining the menu hierarchy as simply as possible using a nest view. Q3 has been extracted from the Android guideline [24], which proposes contextual actions, such as actions that affect a specific item or context frame in the UI. This guideline describes different alternatives to display contextual action items. With these three questions, we began to define a part of a branch in our tree structure (Figure 6).

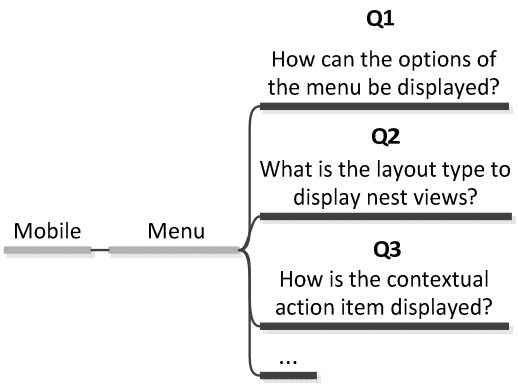


Fig. 6. Example of questions

SE3) Defining the answers. For question Q1, we have identified the alternatives *"Button"* and *"Action Bar"*, since both options are the two posible ways to display the options of a menu. This classification is also used in the guidelines of Symbian [27], iOS [23] and Android [24]. Figure 7 shows an example of button and action bar. The different between them is that the button is based on option displayed by pressing the Buttons while the action bar is based on the combination of on-screen action items and over flow options.

For question Q2, we have identified the alternatives *"Linear"*, *"Relative,* and *"Web view*", which appear in the Android guidelines. These answers gather all the possibilities to display a nest menu. These alternatives are also used in the design guidelines of Symbian and iOS. Figure 8 shows an example of "*linear"*, "*relative"* and "*Web view"*. All of them deal with the arrangement of view hierarchy. *"Linear"* arranges the view in a single column or in a single row. *"Relative"*, arranges the view in sections, and *"Web"* arranges the view as a web view.

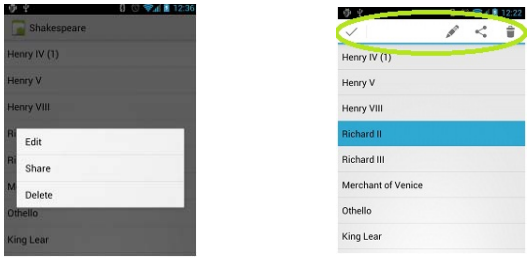
For question Q3, we have identified the alternatives "*Floating contextual"* and *"Contextual action mode"*. These answers have been defined using the design Android guidelines [24],[23] Figure 9 shows an example of a floating contextual menu and a contextual action mode. The difference between both types is that the Floating contextual display actions using a flying list, while the Contextual action mode displays action item on the screen.

a) Button *b) Action Bar*

Fig. 7. Alternatives Design for question Q1

a) Linear *b) Relative* *c) Web view*

Fig. 8. Alternatives Design for question Q2



a) Floating Contextual Menu *b) Contextual Action Mode*

Fig. 9. Alternatives Design for question Q3

Figure 10 shows how the tree is built using the questions and answers identified in our example. Next, we must continue following this procedure in order to define questions and answers until we do not have any more design alternatives defined by interface design guidelines.

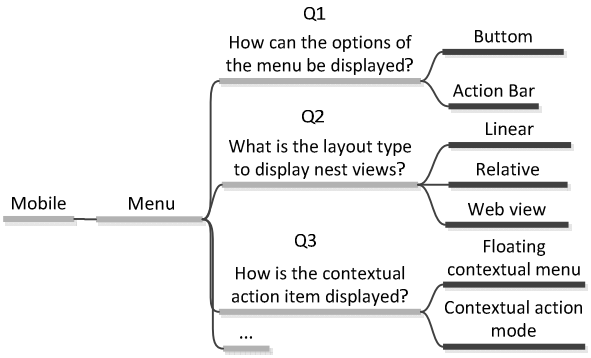


Fig. 10. Example of answers

SE4) Recommending usability guidelines: Following this process, once the answers have been defined, we must define which answers are recommended by usability guidelines.

As shown in Fig 10, for question Q1, two design alternatives (answers) are considered: *Button* and *Action bar*. Their respective recommendations are given with respect to the context of use (type of platform). For example, the alternative *Button* is recommended if we are developing an application for Symbian, Nokia, or Android (lower until version 2.3) platforms. This design alternative fulfils the usability feature which is stated in Nielsen heuristic [33], *"match between system and the real world",* because the user activities should follow real-world conventions without essential changes. The alternative *Action bar* is recommended when the application is planned to be developed for Android (version 3.0 or higher) [24]. This design alternative fulfills the usability feature

"flexibility and efficiency of use" according to Nielsen's heuristics [33]; since it offers flexibility for accessing actions.

For question Q2, three design alternatives are considered: *Linear*, *Relative* and *Web view*. The recommendations are given taking into account all platforms [24], [27], [23] and considering the tasks for which they are used. For example, the alternative *Linear* is recommended when the tasks consists of displaying content that has dynamic layout, or is not pre-determined, or the menu structure is not too deep [24]. This design alternative fulfills the usability feature which is stated in Nielsen heuristic [33], "*give people a logical path to follow*", because the information should appear in a logical order. The alternative *Relative* is recommended when the task is to locate the main actions easily without high hierarchy. This design alternative fulfills the usability feature, "*minimize the user's memory load by making the object, action and option visible*" specified by Nielsen's heuristic [33] since the user does not need to remember information required for her/his activities. The *Web view* alternative is recommended when the task is to embed a web browser into the action. In this case, the design alternative fulfills the usability feature "*Any such information should be easy to search, focused on the user's task*", according to Nielsen's heuristic [33], because frequency actions are tailored by users.

For question Q3, two design alternatives are considered, the *Floating contextual menu* and the *Contextual action mode.* These have been selected for use with Android and Symbian platforms, and tasks in which they are used. We recommend using the *Floating contextual menu* alternative when the task consists of displaying the contextual menu on views displayed by list view or grid view, where the user can perform direct actions on each item. This design alternative fulfills the usability feature "*The main tasks should be available quickly*" recommended by the Symbian usability guideline [27] since the actions frequently used should have priority in terms of visibility. The *Contextual action mode* alternative is recommended when the task is to perform an action on multiple items at once. This alternative fulfills the usability "*The help would assist the user in making full use of the functionalities*" according to Nielsen's heuristic [33], since the user should be informed about what is going on.

The recommendation was continued for each alternative, but the usability guidelines are not always in concordance with the context, task and/ or user; so situations involving contradiction exist. For example, when the task consists of defining the hierarchy of the actions, a recommendation is that the application "Can suffer from poor usability and discoverability" if a drop down is used. This is a piece of advice contemplated in the Symbian platform. When the context is the Android platform, the drop down is called "linear layout", and the advice is to use it when the task is to reduce the hierarchy of views on applications. Therefore, the recommendations have been made according to context, task or user.

SE5) Defining the group of questions: Questions: *Q1, Q2, Q3,* are grouped by "Menu", since the end-user must be asked all of them in order to know the requirements with regard to the menu. We differentiate the group of questions in the tree structure with the character "*", as Figure 11 shows.

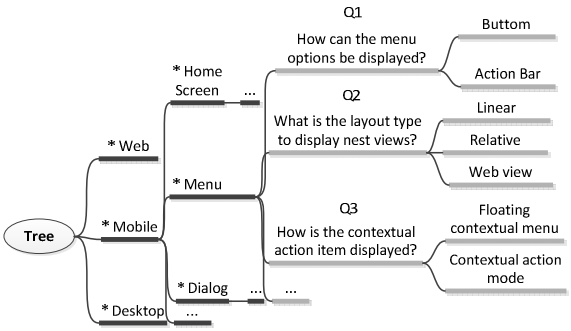
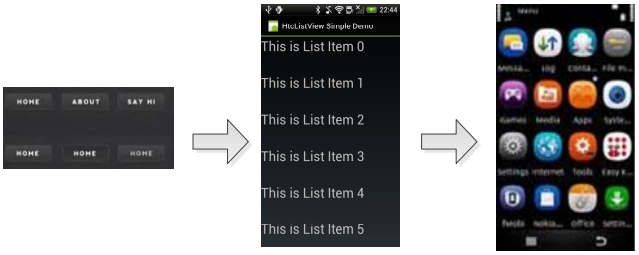


Fig. 11. Example of groups of questions

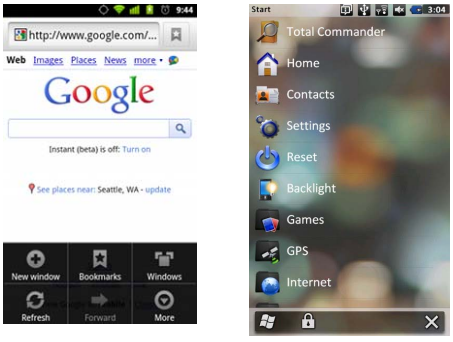
SE6) Obtaining interface designs: At the end of our navigation we arrive at a set of designs depending on the user's requirements. For example in Figure 18, we arrived at the leaf *Grid* following the sequence: Mobile → Menu → *How can the menu options be displayed?* →Button → *What type of menu is required?* → *View menu* → *What is the item display mode* → *Grid* → *Grid View*. Figure 12 shows the differences between the designs of *Button, View Menu* and *Grid.* Depending on the the end-user's answers, the navigation process guides the analyst towards one of these designs.



Button *View Menu* *Grid*

Fig. 12. Sequence of alternatives in order to obtain a design.

As the same way, we could obtain other alternatives of design. Such designs are depicted in Figure 13. These are obtained following the same trajectory but selecting the alternative Six Button or List as answers for question *What is the item display mode?* (See *Q8* in Figure 18)



a) *Six menu button* b) *List view*

Fig. 13. Some posibles design alternatives.

SE7) Transformation rules definitions: in this stage, we must define transformation rules to transform the designs into primitives of a MDD method. In order to facilitate this transformation, we recommend using UsiXML [29] to specify the designs, since there are existing rules to generate primitives for some MDD methods. The definition of these rules is beyond the scope of this paper, but existing rules can be used with our proposal. For example, there is a set of rules to transform UsiXML interface designs into conceptual models of a MDD method called OO-Method [34].

B. *Phase of use.*

Once we have defined the tree structure, we can use it to capture requirements. Figure 18 shows tree structure of our example completed with more questions and answers. The navigation process in the tree starts from the root to the leaves. Next, we describe a possible navigation process to capture the requirements for a mobile phone. Since we are developing for a mobile platform, we start selecting the alternative *Mobile* from the root. Inside *Mobile* there are other groups of questions (*Menu, Dialogue*, among others). The end-user must be asked the questions in all these groups of questions. We begin our navigation process with the first group, *Menu* (GQ1 → GQ2).

Once we begin the flow through the *Menu*, we follow the next sequence of branches:

- The Navigation process derived from Q1. A possible sequence could be*: Q1→GQ3→Q4→GQ5→Q8→A3→ D1*. With this navigation process we can arrive at design D1-*Grid View (*See Figure 14*)*. Once we arrive at a leaf, the navigation process continues with the closest unresolved question. In this example, we must continue with Q9, since it was in a group (GQ5) together with Q8. This navigation process brings us to D2 *(Drop Down Menu)* through *Q9→A5→D2 a*rriving at design D2. Figure 15 shows an example of this design. The flow continues with the other questions in GQ3.

Fig. 14. Design D1 - Grid view.

Fig. 15. Design D2 – Drop Down menu.

- Navigation process derived from Q2: A possible sequence could be: *Q2→A16→D3*. Since A16 was selected, we arrived at design alternative D3. Figure 16 shows a possible design for D3.

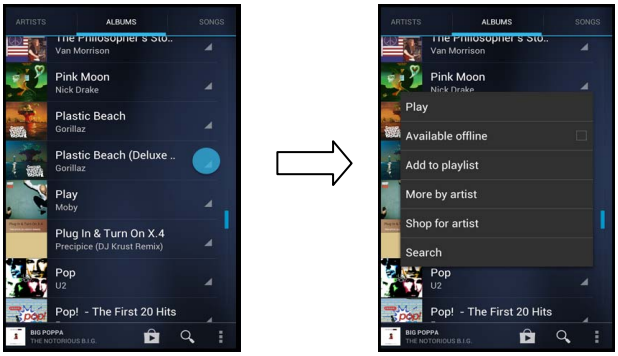


Fig. 16. Design D3 - Linear Vertical with nest view.

- Navigation process derived from Q3: A possible sequence could be: *Q3→A19→D4*. This last selection addresses us to the *Floating Contextual Menu* design, represented by D4 in Figure 18. A possible design for D4 is represented in Figure 17.

Fig. 17. Design D4 - Floating Contextual Menu.

At this point we have ended up with a design that is composed of D1, D2, D3 and D4. These designs will be gathered with the other designs arrived at through the whole navigation process. Finally, the designs arrived at can be transformed into conceptual primitives of an existing MDD method according to previously-defined transformation rules. Note that we have not exemplified this process since these transformations are beyond the scope of the current paper.

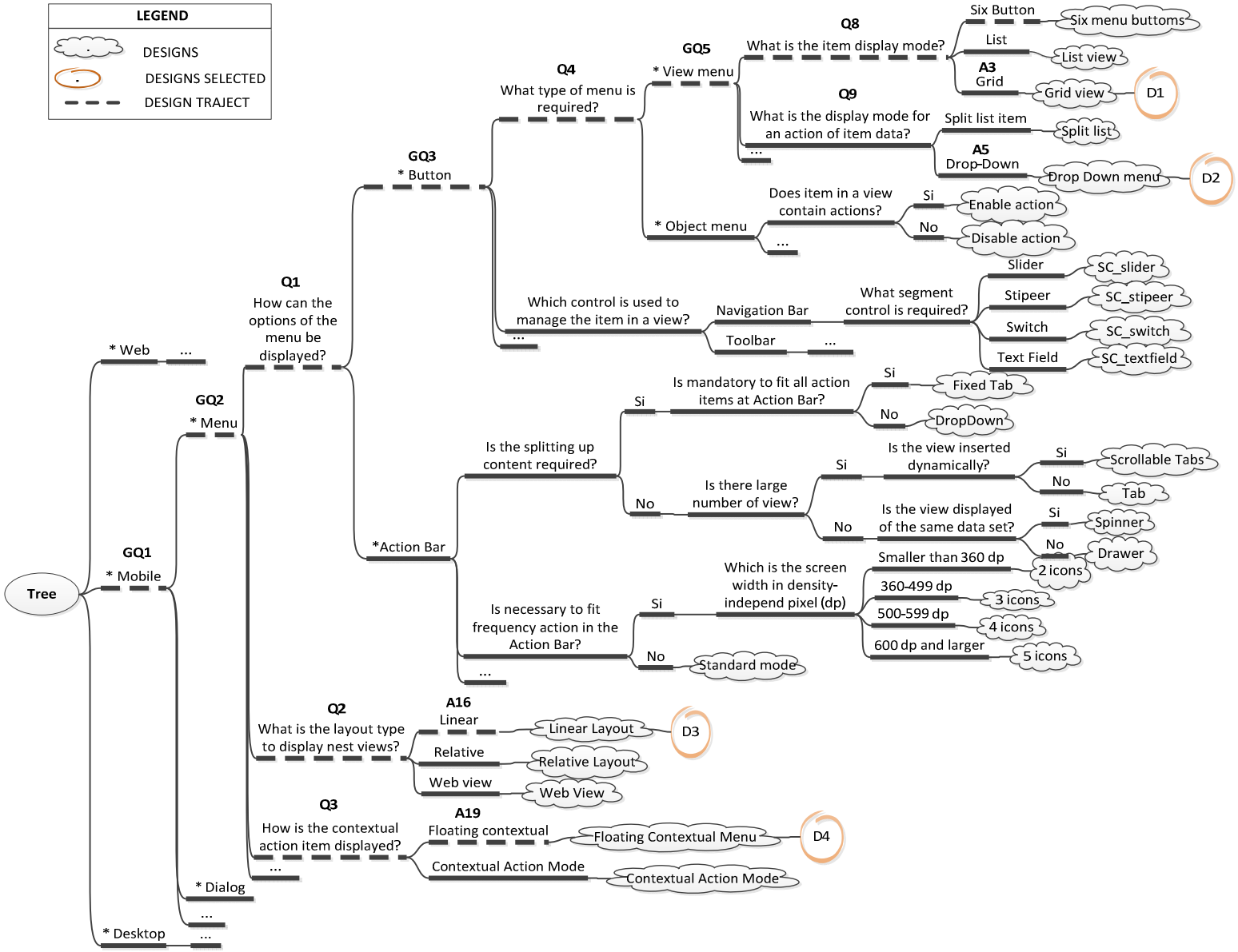


Fig. 18. Usability Requeriments Capture

VI. CONCLUSION

This paper presents an approach to deal with usability requirements in MDD environments. The process consists of building a tree structure using interface design guidelines and usability guidelines that helps the analyst to capture usability requirements. The approach is based on a question-answer format in such a way that requirements are captured with an interview with the end-user. The output of the interview is a set of designs that the system must satisfy. If we specify these designs formally, we can transform them into conceptual primitives of an existing MDD method.

As a language to specify the designs, we recommend UsiXML, since there are current works that have defined transformations between this language and existing MDD methods. However, our proposal is independent of the language to specify the designs. Note that the approach is also independent of the MDD method we used as the target of the transformations. However, if the chosen MDD method does not have conceptual primitives to express interaction features, we could hardly define transformations from the designs to the conceptual model, and few requirements could be included in the software development process. The tree structure and the transformation between the designs and the MDD method are defined once only, and they can be reused indefinitely to develop any system.

Note that the size of the tree structure will increase with the number of guidelines we consider. Even with few guidelines, the size of the tree is difficult to manage if we do not have a tool. As future work, we plan to develop a tool that helps with the definition of the tree structure and with navigation through the branches. In order to simplify the structure, we recommend focusing only on the more frequently used interface design and usability guidelines .

The main contribution of this work is the definition of the process to capture usability requirements, but there is still a lot of work needed to make this viable. The next step is to enrich the existing transformation rules from UsiXML to a MDD method in order to ensure that we can work with any design. Next, with a tool to support the process and the transformation rules, we plan to empirically evaluate the proposal. For this aim, we will compare a software development using our approach to capture usability requirements with a development which does not take these requirements into consideration.

REFERENCES

[1] S. J. Mellor, A. N. Clark, and T. Futagami, “Guest Editors' Introduction: Model-Driven Development,” IEEE Software, vol. 20, pp. 14-18, 2003.

[2] S. Ceri, Fraternali, P., Bongio, A., "Web Modeling Language (WebML): a modeling language for designing Web sites." pp. 137 - 157.

[3] N. Koch, A. Knapp, G. Zhang et al., “Uml-based web engineering,” Web Engineering: Modelling and Implementing Web Applications, pp. 157-191, 2008.

[4] L. Bass, and B. John, “Linking usability to software architecture patterns through general scenarios,” The journal of systems and software, vol. 66, pp. 187-197, 2003.

[5] E. Folmer, and J. Bosch, “Architecting for usability: A Survey,” Journal of Systems and Software, vol. 70, pp. 61-78, 2004.

[6] J. Carroll, M., “Human-computer interaction: psychology as a science of design,” Int. J. Hum.-Comput. Stud., vol. 46, pp. 501-522, 1997.

[7] B. Shneiderman, Plaisant, C., Diseño de Interfaces de Usuario. Cuarta Edicion. Estrategias para una Interacción Persona-Computadora Efectiva, Madrid: Addison Wesley, 2006.

[8] J. Nielsen, Usability Engineering: Morgan Kaufmann, 1993.

[9] E. Dynamic. "UI Styles Guides, " http://www.experiencedynamics.com/science-usability/ui-style-guides

[10] G. Nielsen Norman. "Reports," http://www.nngroup.com/reports/.

[11] S. Cronholm, "The usability of usability guidelines: a proposal for meta-guidelines."

[12] S. Henninger, “A methodology and tools for applying context-specific usability guidelines to interface design,” Interacting with Computers, vol. 12, pp. 225-243, 2000.

[13] L. M. Cysneiros, V. M. Werneck, and A. Kushniruk, "Reusable knowledge for satisficing usability requirements." pp. 463-464.

[14] N. Bevan, "Guidelines and standards for web usability." pp. 22-27.

[15] Y. Pei, and G. Jiao, "The research of Web usability design." pp. 480-483.

[16] J. Nielsen. "Mobile Usability Update ": http://www.useit.com/alertbox/mobile-usability.html.

[17] S. Schneider, F. Ricci, A. Venturini et al., “Usability Guidelines for WAP-based Travel Planning Tools,” Information and Communication Technologies in Tourism 2010, pp. 125-136.

[18] T. Jokela, J. Koivumaa, J. Pirkola et al., “Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone,” Personal Ubiquitous Comput., vol. 10, pp. 345-355, 2006.

[19] A. G. Sutcliffe, S. Kurniawan, and S. Jae-Eun, “A method and advisor tool for multimedia user interface design,” Int. J. Hum.-Comput. Stud., vol. 64, pp. 375-392, 2006.

[20] J. I. Panach, S. España, A. Moreno et al., "Dealing with Usability in Model Transformation Technologies." pp. 498-511.

[21] ISO-9241_11, “Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability,” 1998.

[22] J. Tidwell, Designing Interfaces: O'Reilly Media, 2005.

[23] iOS Human interface Guidelines Apple, 2012.

[24] D. Android, “User Interface Guidelines,” 2012.

[25] N. L. Biggs, "Discrete Mathematics," Oxford University Press.

[26] R. Johnsonbaugh, Discrete Mathematics, Fourth Edition ed., New Jersey: Prentice Hall Intemational, 1997.

[27] Nokia. "Symbian Design Guidelines - Dialogs," http://www.developer.nokia.com/Resources/Library/Symbian_Design_Guidelines/.

[28] L. Cerejo, A. "User-Centered Approach To Web Design For Mobile Devices," http://mobile.smashingmagazine.com/2011/05/02/a-user-centered-approach-to-mobile-design/.

[29] J. Vanderdonckt, Q. Limbourg, B. Michotte et al., "USIXML: a User Interface Description Language for Specifying Multimodal User Interfaces."

[30] M. Maguire, “Context of use within usability activities,” International Journal of Human-Computer Studies, vol. 55, pp. 453-483, 2001.

[31] J. A. T. Hackos, and J. Redish, User and task analysis for interface design: Wiley New York, 1998.

[32] J. I. Panach, Ó. Pastor, and N. Aquino, “A Model for Dealing with Usability in a Holistic MDD Method.”

[33] J. Nielsen. "Ten Usability Heuristics"; http://www.useit.com/papers/heuristic/heuristic_list.html.

[34] J. I. Panach, Ó. Pastor, and N. Aquino, "A Model for Dealing with Usability in a Holistic MDD Method," User Interface Description Language (UIDL), D. F. Adrien Coyette, Juan González-Caballeros, Jean Vanderdonckt. (ed.), ed., pp. 68-77, Lisbon (Portugal), 2011.