QualiHM: A Requirement Engineering Toolkit for Efficient User Interface Design

Mohamed Boukhebouze, Ravi Ramdoyal
and Dimitri Diakodimitris
CETIC Research Center
B-6041 Charleroi, Belgium
Email: {mohamed.boukhebouze, ravi.ramdoyal,
dimitri.diakodimitris}@cetic.be

Ugo Sangiorgi, Mathieu Zen
and Jean Vanderdonckt
Université Catholique de Louvain
B-1348 Louvain-la-Neuve, Belgium
Email: {ugo.sangiorgi, mathieu.zen,
jean.vanderdonckt}@uclouvain.be

***Abstract*—An effective User Interface (UI) is a key success factor for interactive systems. Hence, particular attention should be paid to the UI design during the Requirement Engineering process (RE). Several RE tools have been proposed in order to support the UI design. However, these tools have limitations in terms of requirements completeness, requirements quality analysis and UI generation from requirements. In this paper, we present a new RE toolkit called QualIHM, that deals with the limitations of the existing RE. The toolkit supports the description of requirements in different formats. In addition, QualiHM facilitates the UI design by transforming requirement formats from one to another, generating the UI code and providing feedback about the aesthetic of the UI.**

I. Introduction

Requirements engineering (RE) is the first phase of the software development life cycle that aims to capture, analysis, specify, validate and document stakeholders needs [1]. RE is a crucial activity to better understand these needs and the problem domain, since an inaccurate or wrong requirement can lead to a more costly software development than the original estimation, or to a dissatisfaction of the customer/end-user. Besides, User Interfaces (UIs) are also considered a key aspect of software development, since their effectiveness is pivotal to the success of an interactive system and to maximize user satisfaction. For this reason, UIs are often used as a basis for discussion and validation with the stakeholders during requirements elicitation and analysis [2]. As a result, RE uses many concepts, techniques and concerns of UI design to capture and validate the behavior of the software to be developed.

In the context of requirements engineering, several techniques are used to design an effective UI, such as prototyping, interviews and brainstorming [3]. By using these techniques, requirements can be expressed in different formats: *textual requirement* (e.g. user story or use case); *low-fidelity prototype* (e.g. sketch or wireframe); *high-fidelity prototype* (e.g. widget-based UI) and *model based description* (e.g. use case, task model and domain model). Each of these requirement formats has its importance and benefits within the user interface design process. For example, textual requirements are used to clarify the needs and well-document the domain problem. Low-fidelity prototypes are used to stimulate designers and end-users to discuss and interpret each others ideas [4]. High-fidelity prototypes allow end-users and stakeholders to experiment more interactively the possible future system and to provide additional requirements that need to be implemented [5]. Finally, model-based descriptions can be used to establish a formal understanding about what the software is supposed to do [1].

However, existing academic and industrial RE tools focus only on specific representations of the requirements and do not support the combination of the different requirement formats. Yet, such a combination allows to ensure the completeness, unambiguousness and correctness of software requirements [6]. In addition, existing RE tools do not support the mapping between the requirements formats, which could potentially help keeping trace and consistency between requirements and could favour an efficient UI design.

On the other hand, UI design has become a costly process due to the emergence of new devices with heterogeneous characteristics (e.g. various screen size) and diverse interaction modalities (e.g. tactile and gestural modalities). This heterogeneity should obviously be taken into account during the design of the user interface, but is a difficult and time consuming task. To improve the UI design, several User Interface Description Languages (UIDLs) have been proposed (e.g. UsiXML [7]) to describe UIs independently from any computing platform [8]. These UIDLs rely on Model Driven Engineering (MDE) to describe the UI at different levels of abstraction and transform high-level UI descriptions (e.g. task and domain models) into a UI design model (e.g. concrete UI model) [7]. During this transformation, the context of use can be taken into account in order to generate UIs adapted to the device and the user model. Such UIDLs allow to significantly improve the UI design, yet most existing RE tools do not support them, nor do they provide assistance to help the UI designer to develop a quality user interface, typically in terms of aesthetics.

To deal with these limitations, this paper proposes a new requirement engineering toolkit, called QualiHM (Quality Human-computer Interface Design). This toolkit aims at supporting the requirements gathering through User Interface design by providing four major key features:

- **Description of requirements in different formats**. To ensure the completeness of UI requirements, QualiHM allows to capture textual requirements (through user stories), low-fidelity prototypes (using UI sketches), high-fidelity prototypes and model-based descriptions (through use cases, task models and domain models).
- **Mapping between the requirements formats** . To ensure the traceability of UI requirements, QualiHM enables to define the mapping between the different descriptions of the requirements, hence allowing to link and transform the requirement formats to each others.
- **UIDL support**. QualiHM is compliant with the UsiXML language, which has been chosen due to its expressiveness regarding the description of the different facets of UIs [9]. In addition, UsiXML is in the process of being standardized by W3C.
- **Assistance for UI designers**. The QualIHM toolkit provides assistance to define and ensure the quality of UIs by giving feedback about their aesthetics.

The remainder of the paper is structured as follows. Section II delineates the research background, notably regarding RE and UI design. Section III presents the QualIHM toolkit for the design of quality user interfaces. Section IV then illustrates a possible application of QualIHM through a case study. Section V subsequently describes the related works in the field of UI requirement support and discusses how QualIHM overcomes the limitations of existing tools. Section VI finally concludes this paper.

II. Background

The main motivation behind the QualIHM toolkit stems out of the importance of improving the efficiency UI design. To this end, three major research directions have been identified: How to support different formats of requirements ? How to reap the benefits of UIDLs to improve UI design? And how to assist UI designers to develop quality UIs?

A. User interface design

En effective user interface is a key factor of the success of an interactive system. Hence, particular attention must be paid to the UI design during the RE process. This implies taking into account the following questions: Who are the *users* of the interface? What *tasks* do the users perform using the interface? How does user *interact* with the interface? How should the *interface components* be presented to each user? What *commands* and *actions* should the user be able to perform on the interface? [10]. Although there is no single way to answer these questions, there seems to be at least an agreement on the following core activities [11]: (1) The *Requirements elicitation*, aims at capturing and gathering domain knowledge as well as the stakeholders needs; (2) The *Requirements analysis*, aims at identifying the appropriateness, completeness, quality, and value of a set of requirements; (3) The *Requirements specification*, aims to establish an understanding between different stockholders about who the user interface is supposed to be; (4) The *Requirements validation* aims to demonstrate that the requirement statements meets the intended customer needs; (5) The *Requirements management* aims at monitoring the status of the software requirements based on a predefined procedure by managing the requirements change and traceability. To achieve the RE activities, a wide variety of techniques are usually used, including:

- Traditional techniques, such as existing corporate documents analysis, questionnaires, interviews;
- Group elicitation techniques, such as brainstorming and focus groups;
- Early development techniques, such as Prototyping and Rapid and/or Joint Application Development (RAD/JAD);
- Observation techniques, typically contextual approaches such as stakeholder observation or cognitive techniques such as protocol analysis.

B. The requirement formats

By achieving the RE activities and using the RE techniques, requirements can be expressed in different formats.

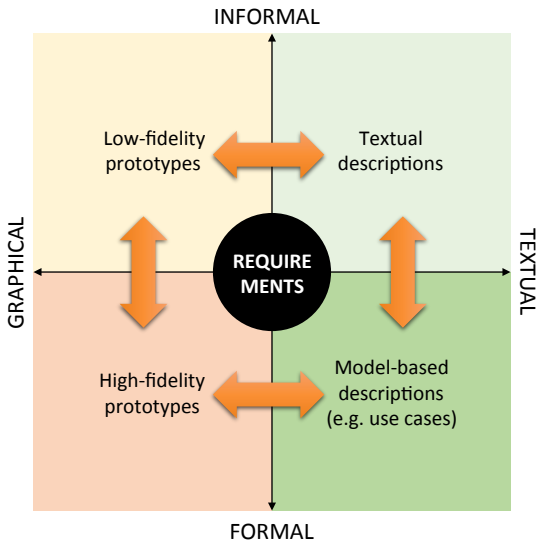

Fig. 1. The different dimensions of the requirement format.

As depicted in Figure 1, requirement formats move along two axis. The *formalisation* axis typically ranges from *informal*, which provides an illustration of the stakeholders needs, to *formal*, which allows to ensure the consistency and completeness checking, mapping support and validation assistance. On the other hand, the *representation* axis typically ranges from *textual*, which enables a textual description, to *graphical* which provides a visual illustration. As result, four typical formats can be considered for requirements:

1) Textual Requirements: Textual requirements consist of informal narrative descriptions of interaction sequences between the users and the interactive system. This requirement format allows to clarify and well-document stakeholders needs. In addition, textual requirements do not require a background or training for producing, understanding, or

using them [12]. Many information can be inferred from textual requirements, especially the functional requirements (the functionalities of the system being developed) as well as non-functional requirements (security, performance, quality, etc.). However, it is hard to identify gaps in a collection of textual requirements, and thus hard to ensure the requirement completeness [13].

2) Low-fidelity Prototype: In UI design, low-fidelity prototypes allow designers to find and eliminate basic problems with the help of end-users at a very early stage of the development cycle, often before any code has been written [14]–[16]. Some studies report that low-fidelity prototypes should solve 80% of the major interface problems [17], with the speed of producing a prototype early (during the requirements-specification phase) outweighing the need to produce a final solution. Low-fidelity prototypes are especially important as tools to test ideas during early design, because they are easy to produce (although hard to replicate), and allow designers to quickly expose problems before committing to decisions [18].

3) High-fidelity Prototype: High-fidelity prototypes are used to provide a UI version that is close to the final version and contains a lot of functional and aesthetic details. This requirement format permits to simulate the final version of the UI in order to allow end-users and stakeholders to gain the experience and to provide other requirements that need to be implemented in the next prototyping [6]. In addition, this format enables the evaluation of the usability of the final UI [19]. However, high-fidelity prototyping is a time-consuming process due to the efforts required to build the prototype. For this reason, high-fidelity prototyping is usually developed without paying attention to the code quality. In this case, the prototype is thrown away after requirements engineering phase, and the final UI is redeveloped from scratch.

4) Model-Based Description: This requirement format uses predefined models to establish a formal understanding of UI requirements [20]. For example, a use case model can be used to formally describe the interactions between the users and the interactive system (e.g. normal sequences, alternative sequences, exceptional behaviour, error handling, etc). Another example is the task model that is used to describe a hierarchy of the tasks (task and the sub-tasks) performed by the users and the system. However, this requirement format requires a background and significant experience for producing, understanding, and using them.

While the formats have individually important roles when addressing different aspects of the requirement engineering process, the use of just one format might not be enough to ensure the completeness, unambiguousness and correctness of requirements. We argue that a combination of multiple formats should be considered when capturing requirements, since it might potentially help on both understanding the users' needs (i.e. the requirements are not ambiguous) and validating the gathered requirements (i.e. the requirements are correct).

C. Using UIDL During The UI Design

In recent years, several User Interface Description Languages (UIDLs) have been emerged for the development of a new generation of the user interfaces that support multi-platform, multi-user and multi-modality [9], [21], [22]. UIDLs describe a user interface independently of any implementation technology. This independence is achieved by relying on Model Driven Engineering (MDE) approach to specify a set of models representing the UI at different levels of abstraction. These languages allow to improve the UI design by defining a single user interface for multiple devices and platforms [8]. In addition, UIDLs permit the reuse of a UI by supporting its evolution, extensibility and adaptability [21]. UsiXML (USer Interface eXtensible Markup Language) is an example of UIDL. This language describes the UI at four main levels of abstractions: task and domain, abstract UI, concrete UI, and final UI. Besides, UsiXML uses a sets of transformations to derive a UI model from another model. For example, a high-level model (e.g. task and domain model) can be transformed into low-level analysis or design model (e.g. concrete UI model) [23]. Another example of a UsiXML transformation is the extraction of high-level model from a set of low-level models or from code [23]. UsiXML is used in different research projects, leading to a standardisation action plan [9]. Nevertheless, such UIDLs are not well supported by existing RE tools.

D. Allowing Assessment of Quality During the UI Design

Some studies reported in the literature indicate that the end-users are strongly influenced by the aesthetics of User Interfaces when using information systems [24]–[26]. According to those studies, the response time for performing tasks is strongly affected by the aesthetic and usability level of the interface [24]. For this reason, it is suitable to evaluate the aesthetic during the UI design phase, aiming primarily at avoiding re-work at later phases. This evaluation can be performed using several techniques including, but not limited to: (1) The visual techniques [27], [28] that involve specific guidelines to analyse the arrangement of the UI components and (2) The aesthetic metrics [29], [30] that rely on mathematical formulas allowing quantification of the UI aesthetic quality (e.g. UI components alignment, color balance). Nevertheless, the existing academic and industrial RE tools do not provide assistance to help the UI designer to perform such evaluation and to give aesthetics recommendations.

III. QualiHM Toolkit Overview

This section presents an overview of the QualIHM (Quality Human-computer Interface Design) requirements engineering toolkit for efficient user interface design. The toolkit ensures the completeness and unambiguity of the requirements by enabling to use different formats to describe the stakeholder needs. In order to favour the consistency and correctness of the requirements, the toolkit provides a mapping between the requirements formats. This mapping consists of the relationship and the transformation between requirement formats.

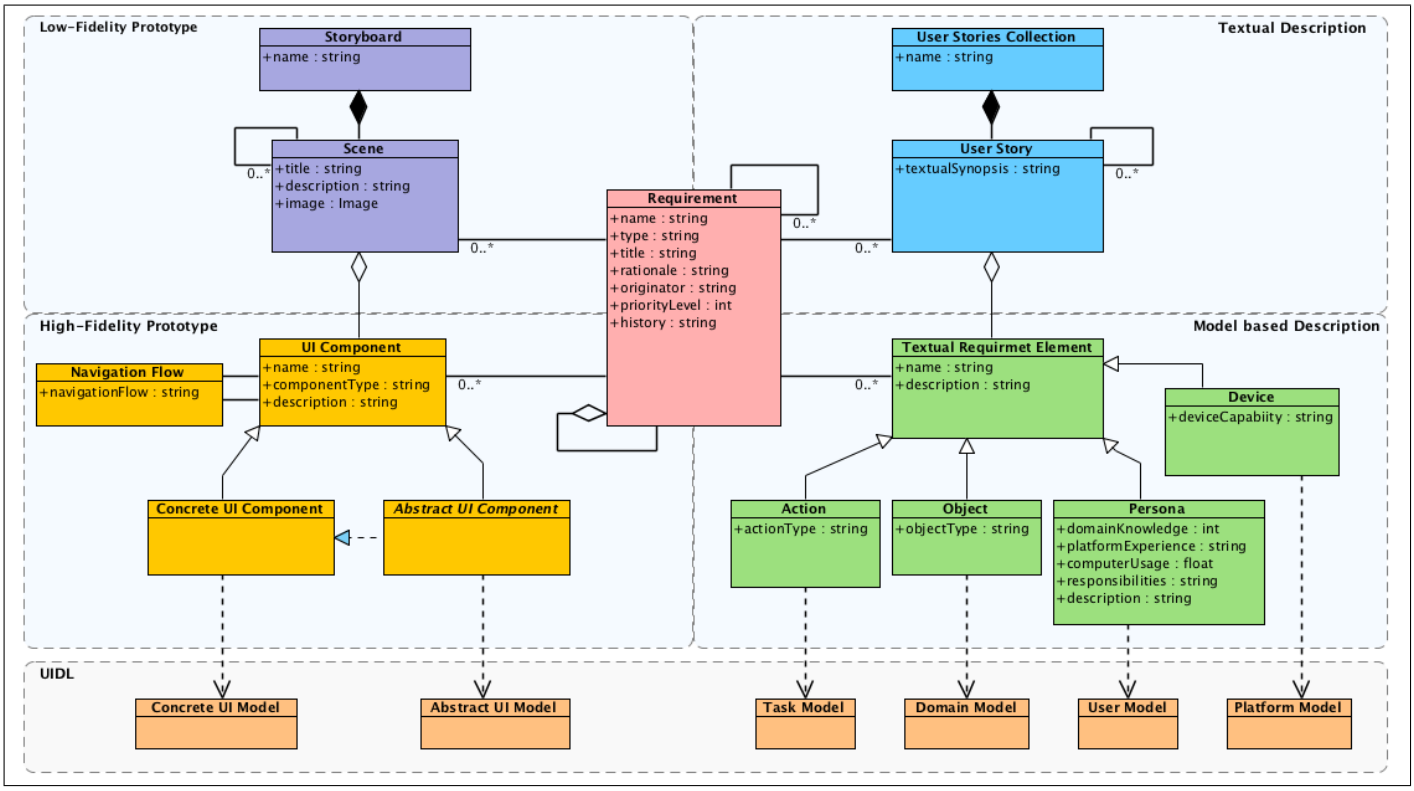

Fig. 2. The key concepts of the QualiHM Toolkit

In addition, QualiHM supports a UIDL-compliant approach allowing the automatic generation of final context-aware UIs. Finally, the toolkit provides assistance during the UI design to help UI designers to develop quality UIs.

A. Key Concepts of QualiHM

The QualiHM toolkit relies on a meta-model to capture the core elements of the requirements, as illustrated in Figure 2. This metamodel explicits the key concepts of QualIHM and their distribution among the different modules of the toolkit.

The *Requirement* is the core concept of the QualiHM meta-model. It encompasses the needs of the stakeholder as well as the information, behaviours, constraints, and capabilities that the solution will need. A requirement can be nested or linked to others requirements. Several properties are used to describe a requirement, such as the *rationale* involved on the requirement and the origin of the requirement (*originator*).

A requirement can be expressed in different formats. As explained in Section II, four dimensions of the requirement formats can considered:

- Textual requirements are described using *User Stories*. A user story consists of a piece of unformatted text explaining in common language how (part of) the interactive system should behave. A story is typically a narrative description of the system from the user perspective [31]. A user story can be linked to other stories and grouped into *User Stories Collections*.
- Model-based descriptions are used to formalise the content of textual stories into a set of *Textual Requirement Elements*. These elements consist of:
  - *Actions* that represent the interactive tasks as viewed by the end users interacting with the system.
  - *Objects* that represent the classes of objects manipulated by a user while interacting with the system.
  - *Personas* that represent the profiles of users who are involved in the system.
  - *Devices* that represent the characteristics of the platform in which the system will be executed.
- Low-fidelity prototypes are described using *Scenes*. A scene consists of a graphical representation of a UI using a mockup drawing, a picture of a screen, or a sketched image. A scene can be linked to other scenes and grouped into *Storyboards*. A Storyboard holds a map of interactors composing an interactive map of scenes.
- High-fidelity prototypes are described using a set of *UI Components*. Two kinds of UI components can be considered:
  - *Abstract User Interface (AUI) Components* describe potential UI elements independently from any interaction modality and any implementation technology . An abstract UI component defines abstract containers and individual components (namely input abstract data compounds, selection abstract data compounds, output abstract data compounds or abstract triggers UI).

– *Concrete User Interface (CUI) Components* describe potential UIs after a particular interaction modality has been selected (e.g., graphical, vocal, multimodal). Such a component allows the specification of the presentation and behavior of a UI with elements that can be perceived by the users.

Ideally, for the sake of completeness and correctness, each requirement should be instantiated in each of these four formats, hence offering different representations accessible to the different stakeholders of a project. However, the toolkit is intended to be as flexible as possible so that users may use it according to their own needs and their working approach (e.g. without using low-fidelity prototypes). It should also be noted that providing these different formats to express requirements implies managing the traceability and consistency of these requirements. For instance, one should be able to trace any user story to its associated model-based descriptions and prototypes (and conversely). The modification of that user story should also appropriately impact the associated model-based descriptions and prototypes.

Besides, to reap the benefits of UIDLs and the MDE approach to improve the user interface design, the QualiHM toolkit is linked to the well defined UsiXML UIDL. This language has been chosen chosen for (1) its good expressiveness to describe the different facets of UIs using an MDE approach; (2) its wide use in different research projects (leading to a standardisation action plan in the context of a European project [9]); (3) the availability of different tools including AUI and CUI designers as well as transformers (e.g. the transformation of AUI to CUI and CUI to final UI code). For this reason, the *Textual Requirement Elements* and *UI Components* are described using it corresponding UsiXML models.

B. QualiHM Architecture

To achieve its objectives, the QualiHM toolkit architecture is made up of several components, as depicted in Figures 3. The toolkit architecture respects the principle of "separation of concerns" by allowing to each component to manage a specific aspect of the requirements. The components of the toolkit include:

- *UsiREQ* (User Story Driven Requirement), which is a textual requirement editor that helps capturing the UI requirements in terms of user stories.
- *GAMBIT* (Gatherings And Meetings with Beamers and Interactive Tablets), which is a UI sketching tool that supports prototyping of UIs [32].
- *UsiXML Model Editors*, which are used to edit the different UsiXML models including: task model, domain model, and user model [9]. These models help to formalise the textual requirement description.
- *UsiXML Transformation Tools*, which can be applied to generate AUIs, CUIs and the code of final UIs from task, domain and user models by using set of transformation rules [7]. The obtained AUI and CUI models, as well as the final UI, can be used as high-fidelity prototypes to help to discussion and validate the requirements with the stakeholders.
- *QUESTIM* (Quality Estimator using Metrics), which allows evaluating the UI quality using aesthetic metrics [33]. The main goal of this tool is to provide UI designers with objective feedback about their design.

Besides, the QualiHM architecture uses an Enterprise Service Bus (ESB) as well as a data management framework in order to guarantee an agile and flexible communication between these tools, while ensuring traceability between the different formats. In this way, the architecture favours the extensibility of the QualiHM toolkit by allowing additional tools to be plugged to the toolkit.

As depicted in Figure 3, the different QualiHM toolkit components allow to support a flexible and iterative RE process. Indeed, the QualiHM components (UsiREQ, GAMBIT, UsiXML tools and QUESTIM) can be used in several ways and support several points of views. From the functional point of view, the toolkit covers all requirement activities including elicitation, prototyping, requirement specification, requirement validation, UI quality evaluation and requirement management. From the toolkit usage point of view, the QualiHM components can be combined together in a flexible way. For example, an analyst can first use UsiREQ to describe the stories. From these stories, a designer can use GAMBIT and QUESTIM to define the storyboard and analyse the UI prototype. The stakeholder can subsequently validate these needs by issuing comments using UsiREQ and GAMBIT. To achieve the validation activity, the stakeholder can use GAMBIT to simulate the behavior and the navigation of the designed prototypes. This iteration can be done as many times as required until the requirements are validated by the customer and agreed upon as the final system specifications. Once the requirement is validated, the user interface of the system can be generated from the requirements using UsiXML tools. Another example of the QualiHM components use regards the fact that analysts and designers can work together in order to define the stories and the UI prototypes using UsiREQ and GAMBIT. The stakeholder can, after that, validate these requirements by providing feedback and comments using UsiREQ and GAMBIT. Once the requirement is validated, an analyst can formalise and generate the documentation of the requirement using UsiREQ.

In order to monitor the requirement activities, the requirements can be managed at any time during the requirement process using the different components of the toolkit. This management consists of the requirements change management and the requirement traceability management. The requirements change management allows controlling the introduced, the changed, and the removed requirements, though the consistency of the different formats associated to a requirement is still currently managed manually. The requirement traceability management allows supporting the mapping between the requirements, while helping to transform and keep trace between requirements formats.

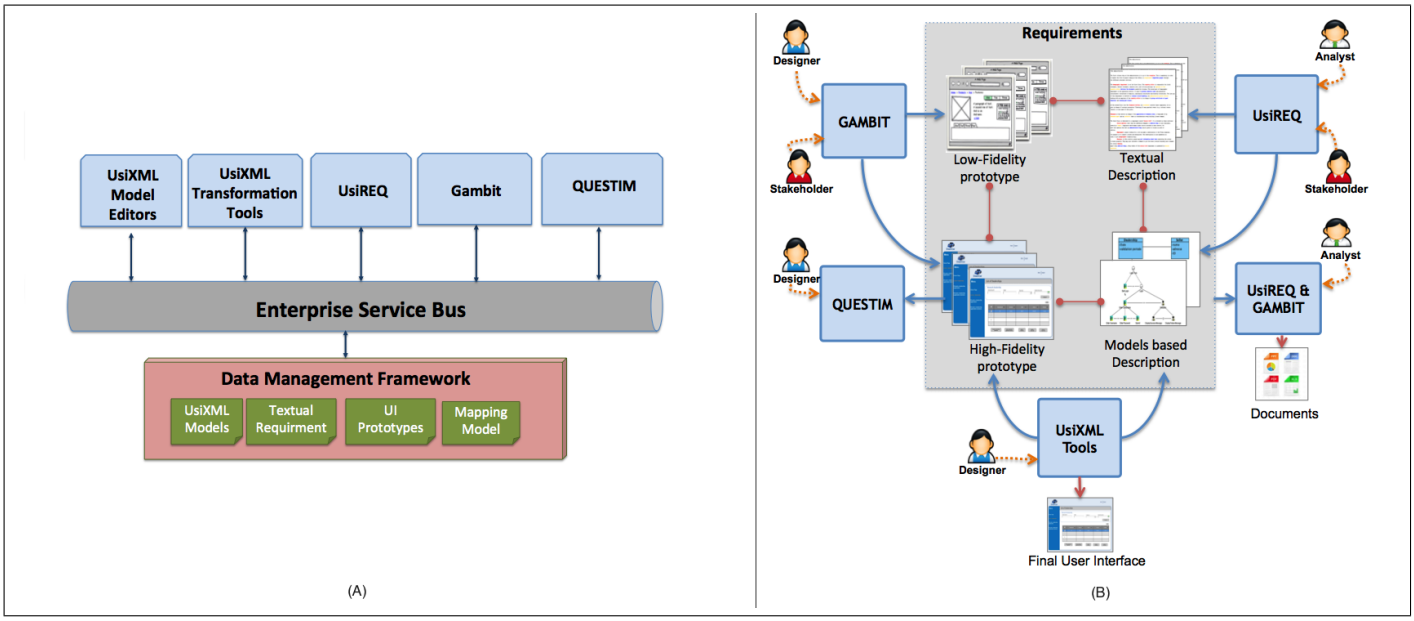

Fig. 3. An overview of the architecture and the RE process of the QualiHM toolkit

IV. User Interface Design with QualiHM

To illustrate how a UI can be efficiently designed using the proposed toolkit, let us consider the following scenario: an automaker company, called DreamCar, designs and manufactures a set of motor vehicles. At the request of car sellers, the company delivers car dealerships to authorise them to commercialise and provide maintenance services for the DreamCar vehicles. To be certified as a DreamCar dealer, a seller should submit an application that demonstrates his/her capacity and experience to lead sales activities. This application is processed by DreamCar to verify if it meets with the company criteria (e.g. application accompanied by a business plan and experience and success in selling car). In order to manage the dealership contracts with theirs car sellers, DreamCar needs to develop a system that allows storing and retrieving the sellers information as well as monitoring the seller applications and dealerships.

In the following, we detail the key features of the QualiHM toolkit using the above scenario.

A. Description of the Requirements in Different Formats

The different components of the QualiHM toolkit allow to capture the requirement in difference formats. For example, textual requirement can be captured using UsiREQ. This tool provides a set of functionalities that enable capturing the UI requirements in terms of user stories. An example of user story that can be expressed using UsiREQ would be: "*In order to encode a new dealership application, the dealership manager should login in to the system. Next, the manager should search the seller who submits the application by providing seller name. When the seller information is displayed, the dealership manager asks the system to encode the new application.*". As showing in Figure 4, UsiREQ eases the writing of such user stories and specifies the properties of this user story (e.g. the title).

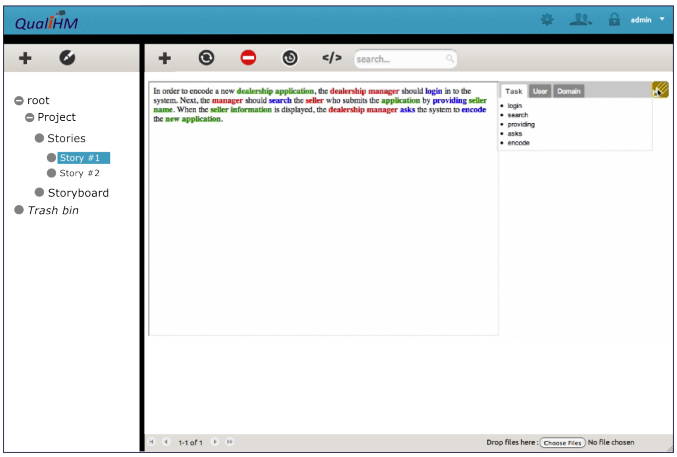

Fig. 4. Capturing the textual requirement using UsiREQ.

The low-fidelity prototyping format can also be developed in QualIHM using GAMBIT. The latter allows to draw and discuss UI prototypes during design sessions or interviews between designers and stakeholders. On one hand, GAMBIT helps designers to express what they understood as systems requirements. On the other hand, the tool allows to the stakeholders and users to have an idea about how the system will perform, what are the interaction flows, etc. To define UI prototypes, users can sketch within scenes and organise the scenes spatially. In the workspace, those objects are represented as rectangles, which can be previously drawn images or produced with the tool. Figure 5 provides an example of a scene of the DreamCar system scenario.

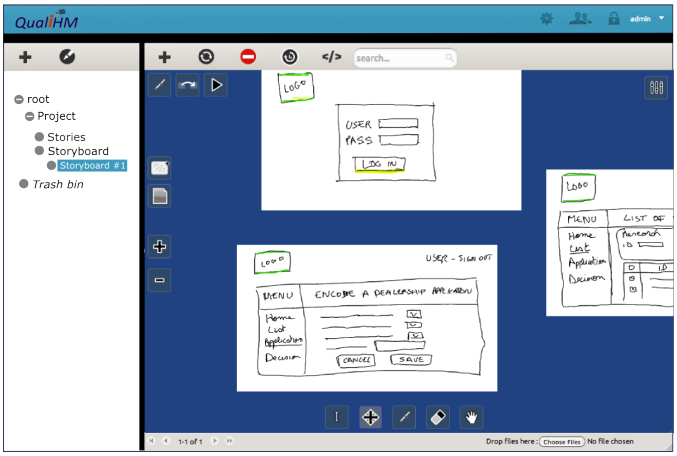

Fig. 5. Low-fidelity prototyping (sketching) using GAMBIT.

B. Mapping Between the Requirement Formats

The QualiHM toolkit allows the definition of mappings between the different requirements formats. This mapping enables the transformation of a requirement format to another format. For example, UsiREQ allows the classification of the terms used to describe the user stories. This classification helps the elicitation of the *Textual Requirement Elements* (task, domain, user models). As depicted in Figure 4, UsiREQ enables the requirement model elicitation by allowing to highlight and derive, from the stories, the concepts that help to develop the software (textual requirement elements). The derived models can be refined using UsiXML editors as explained below.

The mapping also enables the specification of the relationships between the different requirements formats. For example, GAMBIT allows the definition of the navigation between scenes (user interface mockup), as shown in Figures 6. Theses relationships can be used to describe a scenario of the navigation between the scenes so that end-users can discuss and validate the requirements.

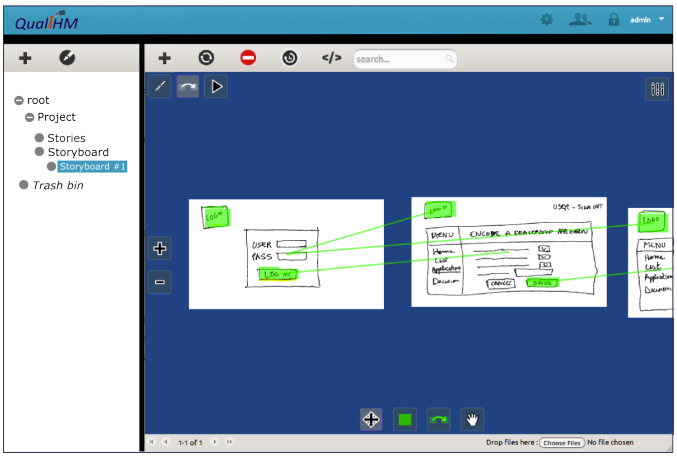

Fig. 6. Mapping between scenes using GAMBIT.

C. Supporting UsiXML language

As explained above, the QualiHM toolkit uses UsiXML models to describe the *Textual Requirement Elements* and *UI Components* in order to benefit from the strength of the UIDL approach. For this reason, the UsiXML model editors are used, within the QualiHM toolkit, to refine the defined or elicited task, domain, user models (see Figures 7). In addition, UsiXML transformation tools can be applied to obtain AUIs, CUIs and final UI code from task, domain and user models. Indeed, task and domain models can be transformed into abstract components using set of transformation rules [7]. For each abstract component, the UsiXML transformation tool creates a concrete component based on the context (user and platform). For example, an output abstract data component can be transformed into a label, which is expressed in the user preference language. Finally, UsiXML transformation tools can be used for the generation of the UI source code from a concrete UI model [7]. Note that, the AUI and CUI models and the final UI can be used as high-fidelity prototypes that help to discuss and validate the requirements with the stakeholders.

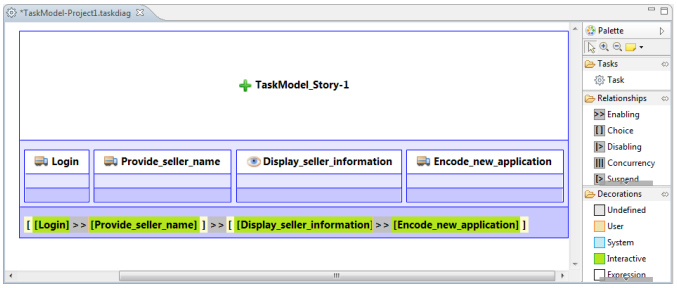

Fig. 7. UsiXML Task Model Editor.

D. Assistance for the User Interface Designer

The QualiHM toolkit allows to assist the designer to build a quality UI by using the QUESTIM tool [33]. The main goal of this tool is to provide designers with an objective feedback about their designed user interface (see Figure 8). Therefore, QUESTIM allows the designer to load a webpage or a UI screenshot and analyze it automatically or semi-automatically by computing metrics based on areas of interest that are drawn afterwards. For each metric result, an interpretation can be defined and a link can be made with the incidence on user preferences and/or performances.

One straightforward way to assess the aesthetics of user interfaces consists in computing metrics that address visual aspects such as Balance, Symmetry and Unity – concepts that are typically considered in aesthetics [29], [30]. QUESTIM tool computes metrics according to different properties (e.g. coordinates, dimensions, colors, etc.) of interest regions defined on top of a UI. Figure 8 shows such interest areas that are defined by the user in order to indicate which parts of the interface are going to be analyzed. Using QUESTIM, Web designers can

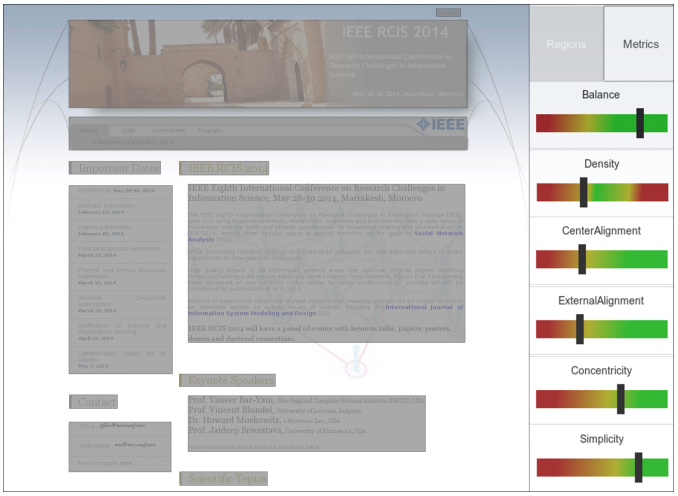

Fig. 8. Evaluation of the user interface quality using QUESTIM.

V. Related Work

This section compare our proposed RE toolkit with the most prominent academic and commercial RE tools.

A. Requirements Engineering Tools

There are many tools related to requirements engineering, both academic and commercial. Basically all tools support some sort of the requirements activities. Hereafter, we present some of RE tools that support the user interface design. Note that, the list of the RE tools considered in this section is not exhaustive. More complete RE tools comparaisons are provided in [34], [35].

Blueprint [36] is a cloud-based solution for collaborative requirements definition and management that is accessible using web-browsers, and targets large distributed enterprise development teams. It provides features to drive a set of requirements to final approval by stakeholders. It manages text requirements, the definition of business processes and domain diagrams, while enabling to create low and high fidelity prototypes. It provides validation mechanisms though use cases and scenarios, with collaborative and traceability features.

Balsamiq Mockups [37] is a prototyping tool that allows users to construct screen mockups by drag-and-dropping and arranging pre-built widgets through an editor based on the "What you see is what you get" (WYSIWYG) principle. The screen mockups can be exported as images.

Just in Mind Prototyper [38] allows users to construct screen mockups and construct prototypes. Also it allows behaviour definition with a visual language that uses widget images, allowing form validation, for instance. Although it allows users to simulate mobile devices, the tool itself only runs on desktop platforms.

Axure RP [39] is a wireframing tool that allows users to construct screen mockups. It can generate interactive HTML website wireframe or UI mockups and allows testing with users. It offers capabilities like drag and drop placement, resizing, and formatting of widgets. In addition, it has features for annotating widgets and defining interactions such as linking, conditional linking, simulating tab controls, show/hide element etc.

IBM's *Rational Rhapsody* [40] helps to analyze and validate requirements, design rapidly with prototypes, and deliver applications using Systems Modeling Language (SysML) and Unified Modeling Language (UML). Rational Rhapsody includes a variety of editions focused on the needs of systems engineers and embedded software developers.

UsiSketch [41] is part of a larger toolkit which includes tools for creating user interfaces with UsiXML. It enables users to sketch user interfaces with different levels of details and support for different contexts of use. The results of the sketching are then analysed to produce interface specifications independently of any context, including user and platform. These specifications are exploited to progressively produce one or several interfaces, for one or many users, platforms, and environments.

The *GUILayout++* tool [42] jointly consider prototyping and evaluation in the development of user interfaces and evaluation criteria for different kinds of prototypes are used. It allows the evaluation of an interface described in UsiXML using metrics, showing percentages of how "pleasant" the interface is. It implements also some aesthetic metrics features as balance, density and uniformity.

The tool proposed by Lemaigre et al. in [43] allows designers to derive a UI model from scenarios, using textual representation for organizing the requirements. Three method levels are successively examined to conduct model elicitation from textual scenarios for the purpose of conducting model-driven engineering of UI's: manual classification, dictionary-based classification, and nearly natural language understanding based on semantic tagging and chunk extraction.

The *RAINBOW* project [44] proposed a tool-supported approach to create prototypical form-based interfaces and semi-automatically analyse them with end-users to derive the structure of the domain model. This user-oriented approach relies on the adaptation and integration of principles and techniques coming from different fields of study, ranging from database forward and reverse engineering to prototyping and participatory design.

B. Comparison of the RE Tools

Table I presents a comparaison of some RE tools with our QualiHM toolkit. This comparaison considers the following criteria:

- Support of the different requirement formats (textual requirement, low-fidelity prototype, high-fidelity prototype and model based description);
- Support of the mapping between the requirement formats by allowing to link and transform the requirement formats to each others;
- Support of the UIDL by describing the UI at different level of abstraction and generating the final user interface;

Tool	Requirement Formats				Mapping		UIDL Support		Assistance
	Textual description	Low-Fidelity Prototyping	High-Fidelity Prototyping	Model based Description	Requirement Linking	Requirement transformation	Abstraction levels description	UI Generation	UI Quality Evaluation
Blueprint	+	+	+	+	+	–	–	–	–
Balsamiq	–	+	–	–	+	–	–	–	–
Just in Mind	–	+	+	–	+	–	–	+	–
Axure RP	–	+	+	–	+	+	–	–	–
Rational Rhapsody	+	–	+	+	+	–	–	–	–
UsiSketch	–	+	–	+	+	+	+	+	–
GuiLayout++	–	+	+	–	–	–	+	+	+/–
Lemaigre's	+	–	–	+	+	+/–	+	+	–
RAINBOW	–	–	+	+	+	–	–	–	–
QualiHM	+	+	+	+	+	+/–	+	+	+/–

TABLE I
THE DIFFERENT TOOLS COMPARED USING THE CONCEPTS IN OUR APPROACH.

- Provide the assistance to evaluate the quality of UIs.

C. Discussion

Table I shows that, each RE tool focuses on a specific representation of the requirements. For example, Balsamiq [37] allows to provide a low-fidelity user interface prototype, unlike *RAINBOW* [44] and *GUILayout++* [42]that allow to provide a high-fidelity UI prototype. In turn, the Lemaigre et al. tool [43] allows to express the requirement in narrative description (textual description), while IBM's Rational Rhapsody [40] supports a model based requirement description (UML). However, unlike QualiHM and Blueprint [36], the RE tools do not support the requirements description of a projet in different formats even if the combination of all requirement formats plays a significant role to improve the user interface design.

Secondly, the majority of the RE tools supports the mapping between the requirements by enabling to link between the requirements. However, many of these RE tools do not support the transformation from a requirement format to another format. Contrary to Axure [39] that enables to transform automatically a low-fidelity prototyping to an high-fidelity prototyping and our proposal toolkit that allows the model elicitation from the stories (like Lemaigre's tool [43]) as well as UI components derivation from scenes (like UsiSketch [41]). This transformation helps to improve the UI design by facilitating the elicitation, specification and validation of the requirements. Note that, in the current version of the QualiHM toolkit, the model elicitation and the UI components derivation are done manually in order to ensure a good precision (see Section IV-B). However, we plan to implement a (semi-)automatic the model elicitation and the UI components derivation in our future work.

Thirdly, QualiHM is a part of the several initiative efforts of academic RE tools to support the UIDL during the UI design. An example of such RE tools is UsiSketch [41], GUILayout++ [42] and Lemaigre's tool [43] that describe user interfaces with UsiXML. This allows to improve the UI design by enabling the definition of a single user interface for multiple devices and platforms as well as generating an context-aware user interface. The QualiHM toolkit was inspired by these tools and supports the UsiXML language (see Section IV-C). However, QualiHM extends these tools by supporting the management of UI requirements as well as the user interface quality evaluation.

Finally, none of the aforementioned tools (except QualiHM and GUILayout++ [42]) help designers to assess the quality of the UI during the design time. QualiHM enables the UI to be evaluated according to a set of metrics. This evaluation can be performed at any time during the UI design: at the elicitation phase (the UI prototypes are evaluated), at specification time (the UI models are evaluated), and at validation time in order to be used as a validation indicator. Note that the current version of the QualiHM toolkit focuses on the static evaluation of UI's and does not support usability evaluation, that will be considered in our future work.

VI. Conclusions and Perspectives

In the context of RE, UIs often serve as a basis for discussion and validation during requirements elicitation and analysis, as well as during the steps of conception, development and testing of a software project. Several RE tools have

been proposed in order to support the UI design. However, these tools have limitations in terms of requirements completeness, requirements quality analysis and UI generation from requirements.

In this paper, we presented the QualiHM toolkit as a RE tool with support for user interface prototyping. This toolkit deals with the limitations of existing RE tools by offering four major key features: (1) the description of requirements in different formats for ensuring the completeness of UI requirements; (2) the definition of the mapping between these requirements formats, as well as the transformation from one requirement format to another; (3) the support of the UIDL approach that enables to support multiple devices and platforms, as well as the generation of context-aware UIs; (4) the evaluation of the quality of UIs by providing the feedback about their aesthetics. The QualiHM toolkit is under LGPL 3 licence. A demonstration video is available at: https://www.youtube.com/watch?v=wzm2zrQcdgs.

In the foreseeable future, we plan to deal with several challenges including:

- Improving the traceability between the requirements: we plan to improve the requirement formats traceability process by allowing a (semi)automatic model elicitation and the UI components derivation;
- Dealing with the consistency: the consistency between the requirement descriptions is an issue of our work. We plan to deal with this issue by implementing a consistency management mechanism. This mechanism should manage the change impact of a requirement;
- Validating our approach: we plan to validate our toolkit using a set of industrial use cases. These use cases will be defined with the collaboration of the industrial sponsors of the QualiHM projects.
- Improving the user interface quality evaluation: we plan to extend the UI quality evaluation by considering, on top of the aesthetic evaluation, the usability evaluation.

ACKNOWLEDGEMENTS

This work was carried out as part of the QualIHM project [45], supported by the *Région Wallonne* and its Collective Research Programme funding (convention number 1217570). We also thanks DefiMedia, industrial sponsor of the project, for its on-the-field expertise and feedback.

REFERENCES

[1] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," in *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: ACM Press, 2000, pp. 35–46.

[2] M. B. Rosson and J. M. Carroll, *Usability Engineering: Scenario-Based Development of Human-Computer Interaction (Interactive Technologies)*. San Diego, CA: Morgan Kaufmann, October 2001.

[3] H. Sharp, Y. Rogers, and J. J. Preece, *Interaction Design: Beyond Human-Computer Interaction*. John Wiley and Sons, 2007.

[4] R. van der Lugt, "Functions of sketching in design idea generation meetings," pp. 72–79, 2002. [Online]. Available: http://doi.acm.org/10.1145/581710.581723

[5] I. Sommerville and G. Kotonya, *Requirements Engineering: Processes and Techniques*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

[6] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[7] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, and V. López-Jaquero, "USIXML: A language supporting multi-path development of user interfaces," *Engineering Human Computer Interaction and Interactive Systems*, pp. 200–220, 2005. [Online]. Available: http://www.springerlink.com/index/barbk4qmlx4ybn8a.pdf

[8] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt, "A unifying reference framework for multi-target user interfaces," *INTERACTING WITH COMPUTERS*, vol. 15, pp. 289–308, 2003.

[9] ITEA2, "UsiXML Project," 2009. [Online]. Available: http://www.usixml.eu

[10] A. Alsumait, "User interface requirements engineering: A scenario-based framework," Ph.D. dissertation, Montreal, P.Q., Canada, Canada, 2004, aAINQ96956.

[11] A. M. Davis and D. Zowghi, "Good requirements practices are neither necessary nor sufficient." *Requirements Engineering*, vol. 11, no. 1, pp. 1–3, 2006.

[12] M. Kamalrudin, J. Grundy, and J. Hosking, "Managing consistency between textual requirements, abstract interactions and essential use cases," in *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*, July 2010, pp. 327–336.

[13] Z. Sharafi, A. Marchetto, A. Susi, G. Antoniol, and Y.-G. Gueheneuc, "An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension," in *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, May 2013, pp. 33–42.

[14] B. Conradi, D. Hausen, F. Hennecke, M. Maurer, H. Richter, and A. Wiethoff, "Prototyping: An overview of current trends, developments, and research in Prototyping," University of Munich. Department of Computer Science. Media Informatics Group, Tech. Rep., 2010.

[15] R. Sefelin, M. Tscheligi, and V. Giller, "Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping," *CHI'03 extended abstracts on Human ...*, pp. 778–779, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=765986

[16] J. N. Petrie and K. A. Schneider, "Mixed-fidelity prototyping of user interfaces," Ph.D. dissertation, University of Saskatchewan, 2006. [Online]. Available: http://www.springerlink.com/index/96r7375x01635515.pdfhttp://dblp.uni-trier.de/db/conf/dsvis/dsvis2006.html#PetrieS06

[17] N. Heaton, "What's wrong with the user interface: how rapid prototyping can help," in *Software Prototyping and Evolutionary Development, IEE Colloquium on*, 1992, pp. 7/1–7/5.

[18] G. Johnson, M. D. Gross, J. Hong, and E. Yi-Luen Do, *Computational Support for Sketching in Design: A Review*. Now Publishers Inc., 2008, vol. 2, no. 1. [Online]. Available: http://www.nowpublishers.com/product.aspx?product=HCI&doi=1100000013

[19] D. Engelberg and A. Seffa, "A framework for rapid mid-fidelity prototyping of web sites," pp. 203–215, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=646869.709393

[20] S. Lauesen and O. Vinter, "Preventing requirement defects: An experiment in process improvement," *Requirements Engineering*, vol. 6, no. 1, pp. 37–50, 2001. [Online]. Available: http://dx.doi.org/10.1007/PL00010355

[21] F. Paterno', C. Santoro, and L. D. Spano, "Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments," *ACM Trans. Comput.-Hum. Interact.*, vol. 16, no. 4, pp. 19:1–19:30, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1614390.1614394

[22] oasis, "User Interface Markup Language (UIML)," 2008. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uiml

[23] Q. Limbourg, "Multi-path development of User Interfaces," Ph.D. dissertation, Université catholique de Louvain, Louvain-la-Neuve, 2004. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Multi-path+Development+of+User+Interfaces.#1

[24] N. Tractinsky, A. S. Katz, and D. Ikar, "What is beautiful is usable," *Interacting with Computers*, vol. 13, no. 2, pp. 127–145, 2000.

[25] A. Sonderegger and J. Sauer, "The influence of design aesthetics in usability testing: Effects on user performance and perceived usability," *Applied Ergonomics*, vol. 41, no. 3, pp. 403 – 410, 2010, special Section: Recycling centres and waste handling ? a workplace for employees and users. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003687009001148

[26] C. Salimun, H. C. Purchase, D. R. Simmons, and S. Brewster, "The effect of aesthetically pleasing composition on visual search performance," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, ser. NordiCHI '10. New York, NY, USA: ACM, 2010, pp. 422–431. [Online]. Available: http://doi.acm.org/10.1145/1868914.1868963

[27] J. Vanderdonckt, "Visual design methods in interactive applications, chapter 7," in *In Albers, M. & Mazur, B. (Eds.), Content and Complexity: Information Design in Technical Communication. Mahwah: Lawrence Erlbaum Associates*, 2003, pp. 187–203.

[28] F. Montero, V. Manuel, and L. Jaquero, "GUILayout ++ : Supporting Prototype Creation and Quality Evaluation for Abstract User Interface Generation," no. June, pp. 39–44, 2010.

[29] J. G. B. David Chek Ling Ngo, Lian Seng Teo, "A mathematical theory of interface aesthetics," *Visual Mathematics*, no. 8, pp. 0–0, 2000. [Online]. Available: http://eudml.org/doc/256723

[30] S. González, F. Montero, and P. González, "BaLOReS," pp. 1–2, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2379636.2379645

[31] M. Rees, "A feasible user story tool for agile software development?" in *Software Engineering Conference, 2002. Ninth Asia-Pacific*, 2002, pp. 22–30.

[32] U. B. Sangiorgi, F. Beuvens, and J. Vanderdonckt, "User Interface Design by Collaborative Sketching," in *Proceedings of the Designing Interactive Systems Conference - DIS '12*. Newcastle Upon Tyne, United Kingdom: ACM Press, 2012, p. 378. [Online]. Available: http://dl.acm.org/citation.cfm?id=2318013http://dl.acm.org/citation.cfm?doid=2317956.2318013

[33] M. Zen, "Metric-based evaluation of graphical user interfaces: model, method, and software support," *Proc. 5th ACM SIGCHI Symp. Eng. Interact. Comput. Syst.*, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2480331

[34] J. M. Carrillo De Gea, J. NicoláS, J. L. FernáNdez AlemáN, A. Toval, C. Ebert, and A. VizcaíNo, "Requirements engineering tools: Capabilities, survey and assessment," *Inf. Softw. Technol.*, vol. 54, no. 10, pp. 1142–1157, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2012.04.005

[35] volere, "Volere Requirements Resources," 2014. [Online]. Available: http://volere.co.uk/tools.htm

[36] Blueprint Software Systems, "Blueprint," www.blueprintsys.com. [Online]. Available: www.blueprintsys.com

[37] Balsamiq Studios, "Balsamiq mockups," www.balsamiq.com. [Online]. Available: www.balsamiq.com

[38] Justinmind, "Justinmind Prototyper," www.justinmind.com. [Online]. Available: www.justinmind.com

[39] Axure, "Axure RP," www.axure.com. [Online]. Available: www.axure.com

[40] IBM, "Rational Rhapsody," http://www-142.ibm.com/software/products/us/en/ratirhapfami/. [Online]. Available: http://www-142.ibm.com/software/products/us/en/ratirhapfami/

[41] Universit catholique de Louvain, LILAB, "UsiSketch," www.LILAB.isys.ucl.ac.be/groups/usisketch. [Online]. Available: www.LILAB.isys.ucl.ac.be/groups/usisketch

[42] F. Montero and M. Jaquero, "GUILayout++: Supporting Prototype Creation and Quality Evaluation for Abstract User Interface Generation," in *Proc. of 1st Int. Workshop on User Interface Extensible Markup Language UsiXML2010 (Berlin)*, 2010.

[43] C. Lemaigre, J. G. García, and J. Vanderdonckt, "Interface Model Elicitation from Textual Scenarios," in *Proceedings of the 1st TC 13 Human-Computer Interaction Symposium (HCIS 2008)- IFIP 20th World Computer Congress, September 7-10, 2008, Milano, Italy*. Springer, 2008, pp. 53–66.

[44] R. Ramdoyal and J.-L. Hainaut, "Involving end-users in database design - the rainbow approach," *IADIS International Journal on Computer Science and Information Systems (IJCSIS), Special issue on "Users and Information Systems"*, vol. 6, no. 2, pp. pp 1–23, October 2011.

[45] Center of Excellence in Information and Communication Technologies (CETIC) and Université Catholique de Louvain (UCL), "QualIHM, a Tool-Supported Methodology for the Quality of Human-Computer Interfaces," www.cetic.be/QualIHM, 2012. [Online]. Available: www.cetic.be/QualIHM