Dealing with Usability in Model-Driven Development Method

Lassaad Ben Ammar(✉), Abdelwaheb Trabelsi, and Adel Mahfoudhi

University of Sfax, ENIS, CES Laboratory,
Soukra Road km 3,5, B.P: 1173-3000, Sfax, Tunisia
{lassaad.ben-ammar,adel.mahfoudhi}@ceslab.org,
abdelwaheb@gmail.com

Abstract. Usability is crucial for the acceptance of Interactive Systems (IS) by end users. Unusable User Interfaces (UI) are probably the main reason why IS fail in actual use. This can explains the increasing number of Usability Evaluation Method proposed in the literature. However, most of these methods are focused on the final product which greatly reduced the ability to go back and makes major changes. Recently, and due to the increasing interest in Model Driven Engineering (MDE) paradigm, the conceptual models have become the backbone of the IS development process. Therefore, evaluating the usability from the conceptual models would be a significant advantage with regard to saving time and resources. The present chapter proposes an early usability evaluation method that is based on conceptual models. The usability evaluation can be automated taking as input the conceptual models that represent the system abstractly. As an output it provides a usability report which contains the detected usability problems. The usability report is analyzed in order to identify the source of problems and suggest changes in the development process.

Keywords: Conceptual model · Model Driven Engineering · Early usability evaluation

1 Introduction

Usability is a quality attribute that has been pointed out as being one of the most important factors in the acceptance of Interactive Systems (IS) by end users. It denote the extent to which specific users can use a product to efficiently and effectively achieve specific goals in a specific context of use [1]. For many years, usability has been perceived as related to the User Interface (UI) [2]. Consequently, usability evaluation is considered late at the development process once the UI is implemented. At this stage, the ability to go back and makes major changes in the design is greatly reduced.

Recently, it has been suggested that usability should be performed from the beginning of the development process in order to increase user experience and

© Springer International Publishing Switzerland 2014
S. Hammoudi et al. (Eds.): ICEIS 2013, LNBIP 190, pp. 405–420, 2014.
DOI: 10.1007/978-3-319-09492-2_24

decrease maintenance costs [3–5]. This has motivated the proposition of methods for the *early usability evaluation*. Among these methods, those following the Model-Driven Engineering (MDE) approach seem to be the most appropriate for the early usability evaluation. In a MDE approach, conceptual models are a primary artefact in the development process. The UI code is (semi-) automatically generated by means a model compiler that takes the conceptual model as input. Hence, evaluating the usability from the conceptual models would be a significant advantage with regard to saving time and resources.

The analysis of some research works that deal with usability in an MDE method highlight some gaps. The main limitation is the lack of precise details given (selection of attributes, scores interpretation, etc.) which makes difficult to understand how these approaches could work correctly in practical settings.

This chapter aims to delineate a method for evaluating usability from the early stage of an MDE method by defining metrics for conceptual primitives that constitute conceptual models. The proposed method can be applied automatically taking the conceptual model, that represents a system abstractly, as input. It is based on the usability model presented in [6] which is empirically validated.

The proposed method focuses on the Cameleon reference framework presented in [7] which structure the UI development interface on four level of abstraction, starting from task specification to a running interface: Task and Concepts, Abstract User Interface (AUI) Concrete User Interface (CUI), and Final User Interface (FUI). We have selected the Cameleon framework because it provides a unifying MDE framework for the development of UI that can be able to adapt to the context of use while preserving usability. Such UIs are namely *Plastic*. In this framework, focus is generally placed on data and functional modeling, disregarding usability aspects. Therefore, there is a need to extend the Cameleon framework in order to promotes usability as a first class entity in the development process.

We structure the remainder of this paper as follows. While Sect. 2 presents an outline of the usability evaluation methods quoted in the literature, Sect. 3 provides a description of our proposed method for early usability evaluation. A case study is presented in Sect. 4 in order to show the usefulness of our proposal to the uncovering of potential usability problems. Finally, Sect. 5 presents the conclusion and provides perspectives for future research work.

2 Related Works

Usability evaluation is often defined as methodologies for measuring the usability aspects of a user interface and identifying specific problems [8]. There exist several methods targeting the usability evaluation of user interfaces. In this section, we focus our interest in the analysis of model-based methods since our main motivation is to integrate usability issues into a model driven development approach.

The usability evaluation has attracted the attention of both Human Computer Interaction (HCI) community and Software Engineering (SE) communities.

The SE community proposed a quality model in the ISO/IEC 9126-1 standard [9]. In this model, usability is decomposed into *Learnability*, *Understandability*, *Operability*, *Attractiveness* and *Compliance*. However, the HCI community has shown in the ISO/IEC 9241-11 standard [1] how usability can be measured in term of *Efficiency*, *Effectiveness* and *User Satisfaction*. Although both standards are useful, they are too abstract and need to be extended or adapted in order to be applied in a specific domain.

Some initiatives to extend both standards are proposed over the last few years. Seffah et al. [10] analyzed existing standards and surveys in order to detect their limits and complementarities. Moreover, the authors unify all these standards into a single consolidated model called Quality in Use Integrated Measurement QUIM. The QUIM model includes metrics that are based on the system code as well as on the generated interface. This makes the application of the QUIM to a model driven development process difficult.

Abrahão and Insfrán [3] proposed an extension of the ISO/IEC 9126-1 usability model. The added feature is intended to measure the user interface usability at an early stage of a model driven development process. The model contains subjective measurement which raises the question about its applicability at the intermediate artifacts. Besides, it lacks of any detail about how various attributes are measured and interpreted.

The usability of a multi-platform user interface generated with an MDE approach is evaluated in [11]. The evaluation is conducted in term of effectiveness, efficiency and user satisfaction. The usability evaluation is based on the experiments with end-users. This dependency is incompatible with an early usability evaluation.

Panach et al. [12] proposes an early usability measurement method. The usability evaluation is carried out early in the development process since the conceptual model. The main limitation of this proposal is that metrics are specific to the OO-method [13]. Therefore, they cannot be applied to other method, which is a disadvantage. They need some adaptation in order to be used (adopted) in other similar methods.

The analysis of the related works allows us to underline some limitations. The majority of the existing proposals lack of guidelines about how usability attributes and metrics are measured and how to interpret their scores. Besides, usability has not been defined in a consistent way across the methods just mentioned. The majority of models includes metrics that are based on the system code as well as on the generated interface which makes their application to a model-driven development process difficult.

It becomes clear that usability evaluation in a MDE approach for the development of UI is still an immature area and many more research works are needed. In order to covers this need, we propose to integrate usability issues into the Cameleon framework. The proposed method is intended to evaluate he usability from the conceptual model. For that reason, we opted for the usability model presented in [6] wherein usability metrics are based on the conceptual primitives.

3 Proposed Usability Evaluation Method

3.1 Overview

The Cameleon framework provides a user interface development process which defines four essential levels of abstraction: Task & Domain, Abstract User Interface (AUI), Concrete User Interface (CUI) and Final User Interface (FUI). The development process takes as input the conceptual models in order to generate the final executable user interface. In this framework, the conceptual models covers the AUI and the CUI levels. The CUI model is the most affected by usability. Therefore, we opted to perform the evaluation from this level. To do that, we proposes a set of usability metrics which are based on the conceptual primitives of this model. The usability evaluation is implemented as a model transformation which takes the CUI model as input and the usability model as a parameter. As outcome, it provides a usability report which contains the detected usability problems. Usability problems are analyzed in order to identify their sources and suggest possible changes in order to improve the usability of future UI obtained as part of the transformation process (see Fig. 1).

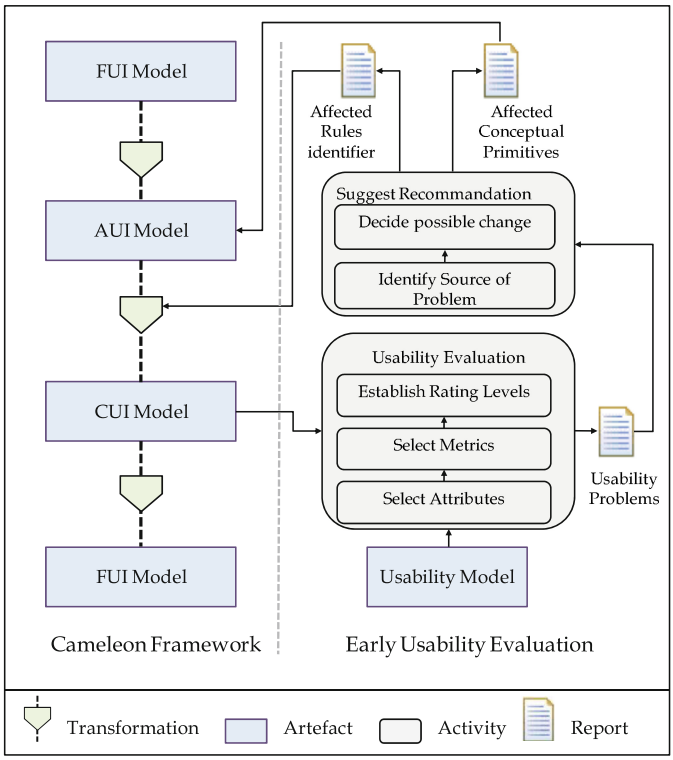

Fig. 1. Incorporating usability into a model-driven development framework.

3.2 Usability Evaluation

Usability evaluation is a process that entails three main steps: (1) Select Attributes, (2) Select Metrics and (3) Establish Rating Levels. In what follows, we briefly describe each step.

Select Attributes. The first step in a usability evaluation method is usually to identify a set of usability attributes. Usability attributes can be either *External* or *Internal*. Attributes which can be measured once the system is implemented are called *External*. Attributes which can be measured before implementing the system are called *Internal*. In this chapter, we focus our interest in the identification of a set of internal attributes. The main objective is to make possible the evaluation without requiring the system implementation. Besides, measuring internal usability can lead to a full automation process.

For the *Learnability* sub-characteristic, which refers to the attributes of a software product that facilitate learning, it can be measured in terms of Prompting, Predictability and Informative Feedback. All these attributes are closely related to the user characteristics. For example, a novice user need to be guided throughout the entire application. Hence, prompting and feedback are essential mechanisms to help user to easily learn the application.

The *Understandability* sub-characteristic refers to the attributes that facilitate understanding. We propose four attributes in order to be able to measure this sub-characteristic. The first attribute is the Information Density which is the degree in which the system will display/demand the information to/from the user in each interface. The Brevity focus on the reduction of the level of cognitive efforts of the user (number of action steps). The short term memory capacity is limited. Consequently, shorter entries reduce considerably the probability of making errors. Besides, the Navigability pertains to the ease with which a user can move around in the application. Finally, Message Concision concerns the use of few words while keeping expressiveness in the error message. Some of these attributes are closely related to the platform features. For example, the screen size has strong influences to the information density and the navigability attributes. Other are related to the users' capacity such as the short term memory.

The *Operability* sub-characteristic includes attributes that facilitate the user's control and operation of the system. Attributes such as Explicit user action, User Operation Cancellability and User Operation Undoability can be used to measure the degree of control that users have over the treatment of their actions. The Error Prevention attribute refers to the means available to detect and prevent data entry errors, command errors, or actions with destructive consequences. The screen size of the platform being used can affect this control when it does not allow displaying button like undo, cancel, validate, etc.

The *Attractiveness* sub-characteristic includes the attributes of software product that are related to the aesthetic design to make it attractive to user. We argue that some aspect of attractiveness can be measured with regard to the Font Style Uniformity and Color Uniformity. The Consistency measure the maintaining of

the design choice to similar contexts. The user preferences in term of color or font style are related to the attractiveness attributes. It should be noted that some environment features (e.g. indoor/outdoor, luminosity level) affect the choice of the color in order to obtain a good contrast which give more clear information.

Figure 2 shows an overview of our proposal for attributes specification.

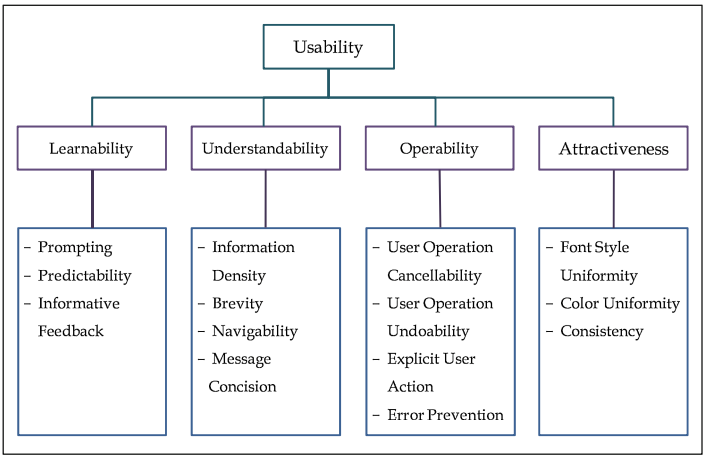

Fig. 2. The proposed usability model.

Select Metrics. All the attributes presented in the previous Section are abstract and can not be measured directly. They need to be more decomposed and detailed in order to be quantified. In order to be able to measure the internal attributes we need to define some *metrics* for each attributes. It should be noted that metrics are intended to measure the internal usability from the conceptual models. Hence, metrics are defined based on the conceptual primitives that constitute the conceptual models of a specific MDE method. The applicability of our method is shown in the present chapter through the MDE method presented in [14]. The main reason of the choice of such method is its compliance to the Cameleon framework and the use of the BPMN notation to describe the user interface models. The BPMN notation is based on the Petri networks which allows the validation of metrics. In what follows, we list the definition of some examples of these metrics. We listed later the indicators defined for each one of the following metric:

– Information Density: The Information Density of a user interface can be measured in terms of the number of the components per user interface. we propose four metric to measure this attribute: average of input element per UI, average of action element per UI, navigation element per UI and total of element per UI.

- Brevity: The human memory capacity is so limited. Hence, each task that require several step will decrease the user satisfaction. The user interface should allow users to accomplish their tasks in a few number of steps. We propose the number of steps (counted in keystrokes) required to accomplish a task as a metric to quantify the brevity.
- Navigability: The navigability measure the level of facilities that the system will provide to navigate throughout several interfaces. We propose the average of navigation elements per UI as an internal metric to measure the navigability attribute.
- Message Concision: Since the quality of the message is a subjective measure, we propose the number of word as an internal metric to measure the quality of the message.
- Error Prevention: To prevent user against error while entering data, we propose to use a drop down list instead of text field when the input element have a set of accepted values.

Establish Rating Levels. The metrics defined previously provides a numerical value that need to have a meaning in order to be interpreted. The mechanism of indicator is restored in order to reach such goal. It consists in the attribution of qualitative values to each numerical one. Such qualitative values can be summarized in: Very Good (VG), Good (G), Medium (M), Bad (B) and Very Bad (VB). For each qualitative value, we assign a numerical range. The ranges are defined build on some usability guidelines and heuristics described in the literature. Next, we detail the numeric ranges associated with some metrics in order to be considered as a Very Good value.

- Information Density: several usability guidelines recommend minimizing the density of a user interface [15]. We define a maximum number of elements per user interface to keep a good equilibrium between information density and white space: 15 input elements (ID1), 10 action elements (ID2), 7 navigation elements (ID3), and 20 elements in total (ID4) [12].
- Brevity: some research studies have demonstrated that the human memory has the capacity to retain a maximum number of 3 scenarios [16]. Each task or goals requiring more than 3 steps (counted in keystrokes) to be reached decreases usability (Minimal Action MA).
- Navigability: some studies have demonstrated that the first level navigational target (Navigation Breadth NB) should not exceed 7 [17].
- Message Concision: since the quality of the message can be evaluated only by the end-user, the number of the word in a message is proposed as an internal metrics to assess message quality (Word Number WN). A maximum of 15 words is recommended in a message [12].
- Error Prevention: The system must provide mechanisms to keep the user from making mistakes [18]. One way to avoid mistakes is the use of ListBoxes for enumerated values. Panach et al. [12] recommend at least 90 % of enumerated values must be shown in a ListBox to improve usability (ERP).

Metrics which are extracted from the proposition of [12] are extracted with their ranges of values. In fact, this ranges are empirically validated. For the others metrics, the ranges of values to consider the numeric value as Very Good are taken into consideration in order to estimate the value to be considered as Very Bad. The Medium, Bad and Good values are equitably distributed once we have the two extremes. The Table 1 shows the list of indicators that we have been defined.

Table 1. Proposed indicators.

Metric	VG	G	M	B	VB
ID1	$\prec 15$	$15 \leq \text{ID1} \prec 20$	$20 \leq \text{ID1} \prec 25$	$25 \leq \text{ID1} \prec 30$	$\text{ID1} \geq 30$
ID2	$\prec 10$	$10 \leq \text{ID2} \prec 13$	$13 \leq \text{ID2} \prec 16$	$16 \leq \text{ID2} \prec 19$	$\text{ID2} \geq 19$
ID3	$\prec 7$	$7 \leq \text{ID3} \prec 10$	$10 \leq \text{ID3} \prec 13$	$13 \leq \text{ID3} \prec 16$	$\text{ID3} \geq 16$
ID4	$\prec 20$	$20 \leq \text{ID4} \prec 30$	$30 \leq \text{ID4} \prec 40$	$40 \leq \text{ID4} \prec 50$	$\text{ID4} \geq 50$
MA	$\prec 2$	$2 \leq \text{MA} \prec 4$	$4 \leq \text{MA} \prec 5$	$5 \leq \text{MA} \prec 6$	$\text{MA} \geq 6$
NB	$\prec 7$	$5 \leq \text{NB} \prec 10$	$10 \geq \text{NB} \prec 13$	$13 \leq \text{NB} \prec 16$	$\text{NB} \geq 16$
WN	$\prec 15$	$15 \leq \text{WN} \prec 20$	$20 \leq \text{WN} \prec 25$	$25 \leq \text{WN} \prec 30$	$\text{WN} \geq 30$

3.3 Automatic Usability Evaluation Process

Conducting the usability measurement manually is a tedious task. That is why we propose to automatize this process by implementing it as a model transformation process. The model transformation process requires two model as input (the user interface model and the usability model) and provides as outcome a usability report which contains the detected usability problem. In the model transformation literature, the definition of the meta-model[1] is a prerequisite in order to formalize the approach and use a model in a productive way. With regard to the usability meta-model, Fig. 3 show the proposed meta-model.

With regard to the usability report, we propose a simple meta-model which explain the usability problem using the following scheme: the *description* of the usability problem, the *related attribute* is the sub-characteristic and attribute in the model that are affected by the usability problem, the *level* of the detected problem and the *recommendation* to solve such problem (Fig. 4).

It should be noted that the usability evaluation is implemented as a model transformation that take the usability model as a parameter to the transformation engine. To do this, we reformulate the parameterized transformation principles initiated by [19].

[1] A meta-model is a language that can express models. It defines the concepts and relationships between concepts required for the expression of the model.

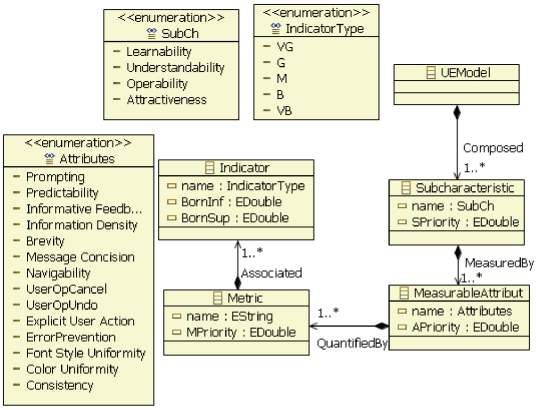

Fig. 3. Usability meta-model.

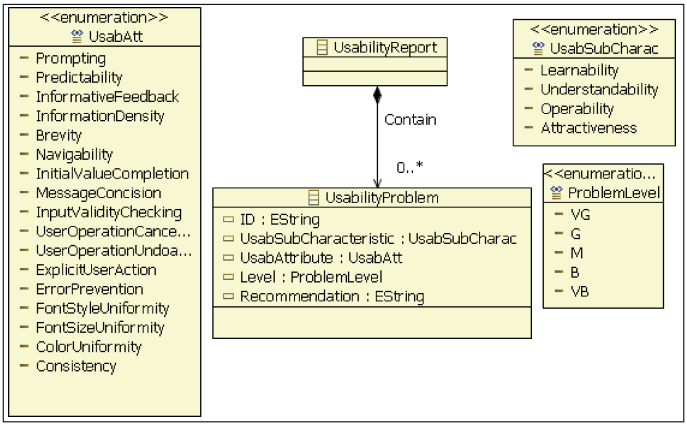

Fig. 4. Usability report meta-model.

3.4 Results Analysis and Suggestion of Recommendations

The main objective of this step is to analysis the *Usability Report* in order to identify the source of the problems and suggest some changes either in the AUI model or in the transformation module. Hence, two reports are produced as an output of this step.

- The first report contains the identifier of the transformation rules that can undergo a slight modification.
- The second report contains the conceptual primitives that have to be modified in order to improve the expressiveness of the AUI model.

As regard to the first report, usability problems that are related to the Brevity and the Information Density attributes usually depend on the transformation rules.

Transformation rules that maximize the Brevity needs to concretize several panels in the some window instead of several window related by a navigation element. Such concretization decrease the number of step (counted in keystroke) required to accomplish a task. Hence, the Brevity is increased. However, the number of widget will be increased which decrease the Information Density. The transformation rule can be modified to concretize several panel with the take into account of the total number of elements which should not exceeds a threshold (e.g., 20 elements per UI).

As regard to the second report, we show a simple example to better explain the impact of the usability problems. There are several way to support the Prompting attribute. Among these way we quote the use of a label that display additional information in order to better guide users while entering data. Additional information may be the required data format or the range of accepted values. The AUI model should provide this information during its transformation. For example, in the underlying method, the UIComponent class should have an attribute (type String) named for example Additional Information. Such attribute contain the additional information to be displayed when it is necessary in order to ensure the prompting.

It should be noted that this step usually requires the intervention of usability experts. We hope that after some experiments we can automatize this step.

4 An Illustrative Case Study

This section investigates a case study in order to illustrate the applicability of our proposal. The purpose is to show the usefulness of our proposal in the assessment of the user interface usability. The research question addressed by this case study is: *Does the proposal contribute to uncover usability problem since the conceptual model?*

The case study is a Requesting a credit to buy a car. The scenario is adapted from [20]. To get information about the credit to buy a car, a bank client does not have to go to the bank branch since the bank portals offer an interactive system which allows resolving such problem. The bank client can perform several tasks using this system: get information about buying tips, simulate a credit to buy a car, request the credit, receive request in line, and communicate with the credit department, etc. For reason of simplicity, we are interested in the <<Request the Credit>> task.

We suppose to have the abstract user interface from Fig. 5 as a result of the transformation of the task model <<Request the Credit>> following the model transformation explained in details in [14]. The result of the transformation is an XML file which is in accordance with the AUI metamodel (left part of Fig. 5). To better clear up the user interface layout, we develop an editor with the Graphical Modeling Framework (GMF) of eclipse. The sketch of the user interface presented by the editor is shown in the right part of Fig. 5.

An ordinary transformation which takes as input the abstract user interface model allows producing the concrete user interface model of Fig. 6. It should

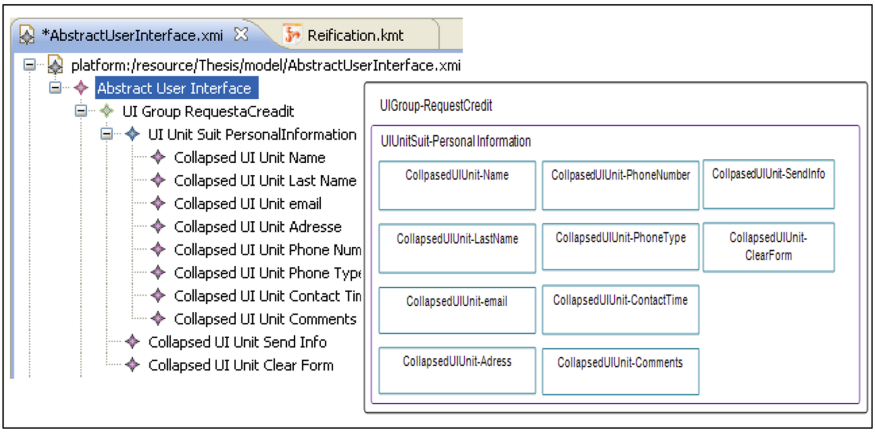

Fig. 5. Abstract user interface.

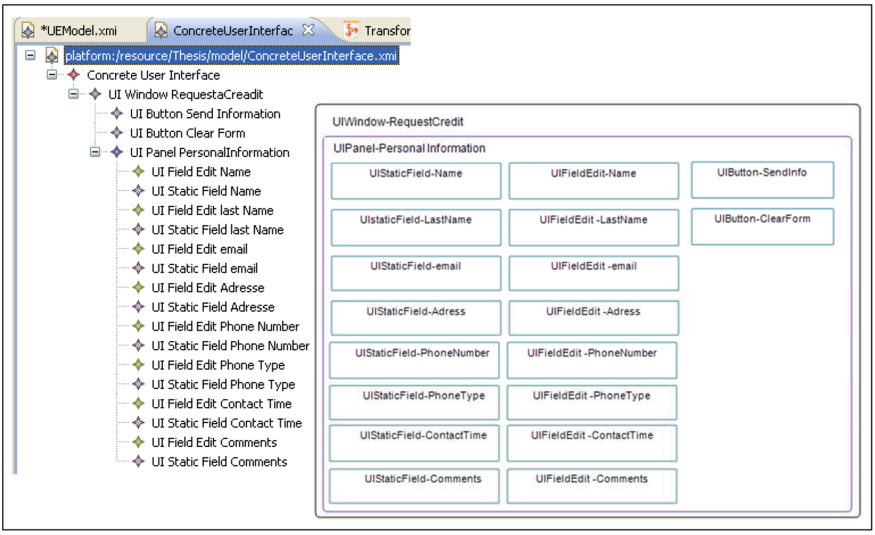

Fig. 6. Concrete user interface.

be noted that this transformation was done taken into account a context of use defined by the analyst. The context is the following: a laptop as an interactive device (normal screen size), an Englishman as a tourist with a low level of experience.

In order to evaluate the concrete user interface, we pursue a reduced version of the usability evaluation process presented in [21]. The *purpose of the evaluation* is to evaluate the usefulness of the proposed model to discover the usability problems presented in the evaluated artifact. The *product part to be evaluated*

is the concrete user interface model. The *selected attributes* are the Information Density and the Error Prevention. The *metrics* selected to evaluate the former attributes are ID2 and ERP. The *indicators* are those presented in Table 1.

The result of the evaluation is a usability report model which contains the detected problems (see Fig. 7).

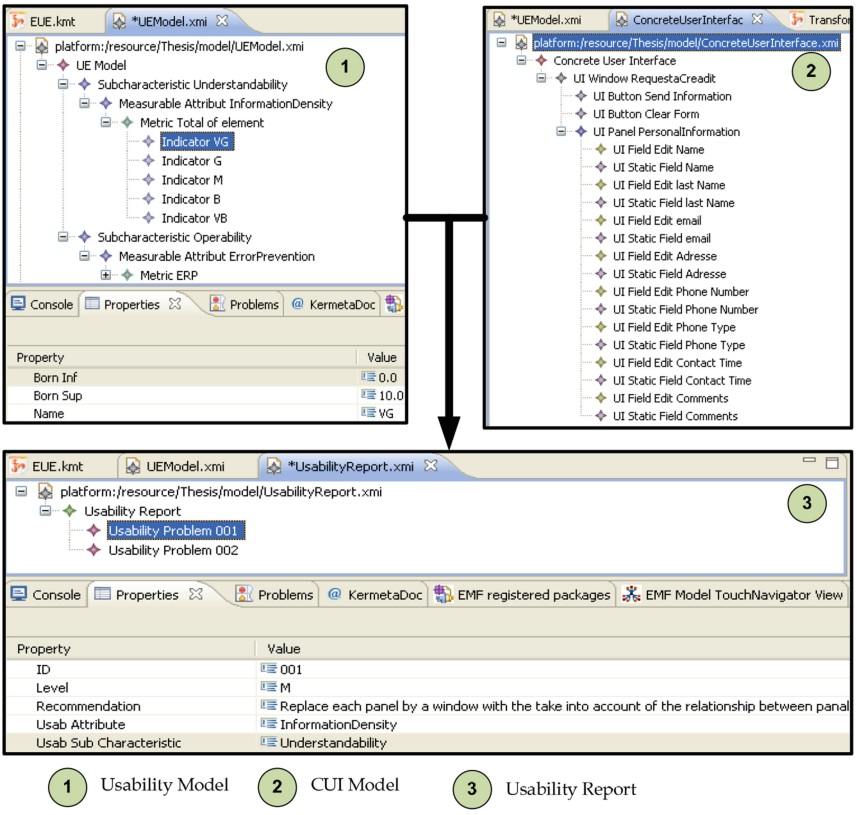

Fig. 7. Automatic usability evaluation.

– Usability problem N1: There are enough elements in the user interface which increase the information density.
Related attribute: Understandability/Information Density
Level: M
Recommendation: Replace each panel by a window with the take into account of the relationship between panel.

– Usability problem N2: There is no means which prevents the user against error while entering data.
Related attribute: Operability/Error Prevention.

Level:B
Recommendation: Each input element with limited values should be displayed in a dropdown list to protect user against error while entering values (e.g. typos).

The second evaluation to be conducted takes an ≪iPAQ Hx2490 Pocket PC≫ as platform. The migration to such platform raises a new redistribution of the user interface elements. The small screen size (240 × 320) is not sufficient to display all information. The number of the concrete component to be grouped is limited to the maximum number of concepts that can be manipulated (5 in the case of <<iPAQ Hx2490 Pocket PC>>). Therefore, the user interface elements are redistributed on several windows (see Fig. 8).

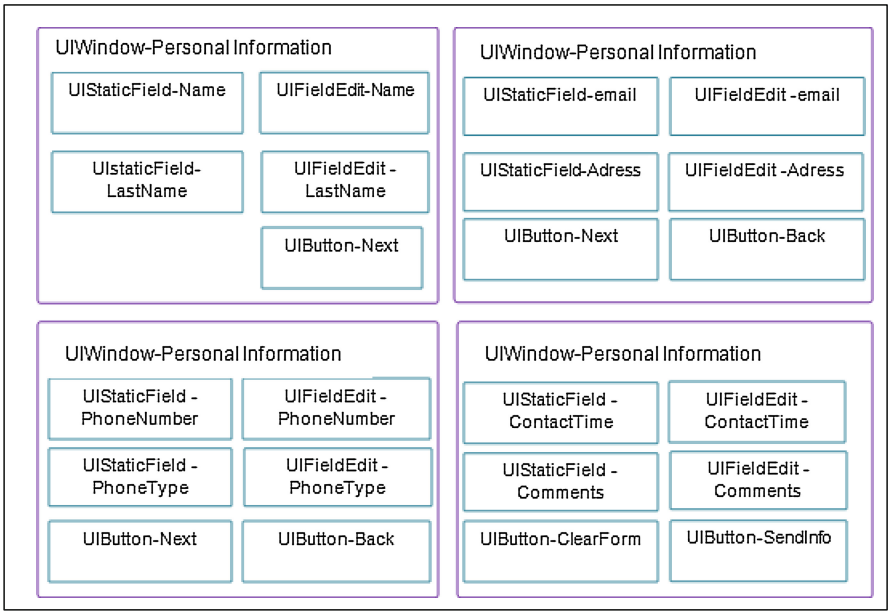

Fig. 8. Concrete user interface for small screen size.

The redistribution of interface elements on several windows will bring more steps to reach the goal. It should be noted that with a small screen size the *Information Density* and the *Brevity* are the most relevant usability attributes to affected. The problem is that these two attributes have a contradictory impact. It is recommended to distribute the concrete components on several screens in order to obtain better *Information Density*. However, redistribute elements from one screen to several will influence negatively the *Brevity* attributes (see Fig. 9).

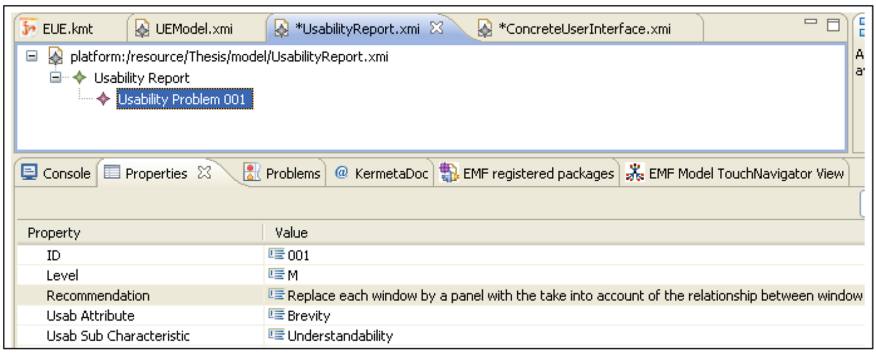

Fig. 9. Usability report.

Learned Lesson
The case study allow us to learn more about the potentialities and limitations of our proposal and how it can be improved. The proposed method allows the detection of several usability problems since the early stage of the development process. The evaluation process may be a means to discover which usability attributes are directly supported by the modeling primitives or to discover limitations in the expressiveness of these artifacts. The ranks of indicators are extracted from existing studies which do not consider the context variation. Therefore, many more experimentations are needed in order to propose a repository of indicators in several cases (medium screen size, small screen size, large screen size). Another important aspect which must be studied is the contradictory affect of usability attributes. For example, the information density and the brevity has a contradictor affect. Increasing the information density will decrease certainly the brevity attribute.

5 Conclusions and Future Research Works

This chapter presents a method for integrating usability issue as a part of a user interface development process. The proposed method extends the Cameleon reference framework. The usability evaluation can be conducted early in the development process (from the conceptual models). The objective is to discover the usability problems presented in the conceptual models. The result of the evaluation is a usability report which contains the usability problems detected during the evaluation step. The analysis of these problems and the identification of their sources allows usability experts to suggest some changes either in the conceptual model or in the model compiler. This can avoid usability problems to occurs in each future UI obtained as part of the development process.

If compared to the existing proposals, our framework presents three main advantages: (1) costs are very low: internal usability evaluation reduce considerably the development costs and maintainability, (2) system does not have to be

implemented, (3) it provides a proper details about how to measure attributes and interpret their scores.

The continuity of our research work leads directly to the investigation of the usability-driven model transformation, the relationship between usability attributes and their contradictory influence to the whole usability of the user interface. The automation of the recommendation step is among the perspective that can be investigated in future works.

References

1. ISO/IEC: ISO/IEC 9241. Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs). ISO/IEC (1998)
2. Seffah, A., Metzker, E.: The obstacles and myths of usability and software engineering. Commun. ACM **47**, 71–76 (2004)
3. Abrahão, S.M., Insfrán, E.: Early usability evaluation in model driven architecture environments. In: QSIC, pp. 287–294 (2006)
4. Molina, F., Toval, A.: Integrating usability requirements that can be evaluated in design time into model driven engineering of web information systems. Adv. Eng. Softw. **40**, 1306–1317 (2009)
5. Panach Navarrete, J.I., Juristo Juzgado, N., Pastor, Ó.: Introducing usability in a conceptual modeling-based software development process. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 525–530. Springer, Heidelberg (2012)
6. Ben Ammar, L., Mahfoudhi, A.: An empirical evaluation of a usability measurement method in a model driven framework. In: Holzinger, A., Ziefle, M., Hitz, M., Debevc, M. (eds.) SouthCHI 2013. LNCS, vol. 7946, pp. 157–173. Springer, Heidelberg (2013)
7. Calvary, G., Thevenin, D.: A unifying reference framework for the development of plastic user interfaces. In: Nigay, L., Little, M.R. (eds.) EHCI 2001. LNCS, vol. 2254, pp. 173–192. Springer, Heidelberg (2001)
8. Nielsen, J.: Usability Engineering. Morgan Kaufmann Publishers Inc., San Francisco (1993)
9. ISO/IEC: ISO/IEC 9126. Software engineering - Product quality. ISO/IEC (2001)
10. Seffah, A., Donyaee, M., Kline, R.B., Padda, H.K.: Usability measurement and metrics: a consolidated model. Softw. Qual. Control **14**, 159–178 (2006)
11. Aquino, N., Vanderdonckt, J., Condori-Fernández, N., Dieste, O., Pastor, O.: Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering. In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '10, New York, NY, USA, ACM, pp. 30:1–30:10 (2010)
12. Panach, J.I., Condori-Fernández, N., Vos, T.E.J., Aquino, N., Valverde, F.: Early usability measurement in model-driven development: definition and empirical evaluation. Int. J. Softw. Eng. Knowl. Eng. **21**, 339–365 (2011)
13. Gómez, J., Cachero, C., Pastor, O.: Conceptual modeling of device-independent web applications. IEEE Multimedia **8**, 26–39 (2001)
14. Bouchelligua, W., Mahfoudhi, A., Mezhoudi, N., Daassi, O., Abed, M.: User interfaces modelling of workflow information systems. In: Barjis, J. (ed.) EOMAS 2010. LNBIP, vol. 63, pp. 143–163. Springer, Heidelberg (2010)

15. M. Leavit, Shneiderman, B.: Research Based Web Design & Usability Guidelines (2006)
16. Lacob, M.E.: Readability and Usability Guidelines (2003)
17. Murata, M., Uchimoto, K., Ma, Q., Isahara, H.: Magical number seven plus or minus two: syntactic structure recognition in Japanese and English sentences. In: Gelbukh, A. (ed.) CICLing 2001. LNCS, vol. 2004, pp. 43–52. Springer, Heidelberg (2001)
18. Bastien, J.C., Scapin, D.L.: Ergonomic criteria for the evaluation of human-computer interfaces. Technical report RT-0156, INRIA (1993)
19. Vale, S., Hammoudi, S.: Context-aware model driven development by parameterized transformation. In: Proceedings of the 1st International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) Held in Conjunction with the CAiSE'08 Conference, pp. 121–133. Springer, Heidelberg (2008)
20. Guerrero, J.: A methodology for developing user interfaces to workflow information systems (2010)
21. Ben Ammar, L., Mahfoudhi, A., Abid, M.: A usability evaluation process for plastic user interface generated with an mde approach. Software Engineering Research and Practice, pp. 323–329. CSREA Press, Las Vegas (2012)