Usability Driven Model Transformation

Lassaad Ben Ammar
University of Sfax, ENIS, CES Laboratory
Soukra km 3,5, B.P: 1173-3000
Sfax TUNISIA
benammar_lassaad@hotmail.com

Adel Mahfoudhi
University of Sfax, ENIS, CES Laboratory
Soukra km 3,5, B.P: 1173-3000
Sfax TUNISIA
adel.mahfoudhi@fss.rnu.tn

***Abstract*—Model Driven Engineering (MDE) is a software approach which promotes the use of models and model transformations as primary artifacts in the development process. Recently, there has been wide interest in applying MDE approach in the Human Computer Interaction (HCI) field. It has been proved that MDE is an appropriate technique to generate as automatically as possible the final user interface from the conceptual model. Given a source model, there may be several ways to transform it into target model. Alternative target models are equivalent from the functional perspective and may differ in their usability attributes. Driven the model transformation process by the usability properties is as yet unexplored territory. This study attempts to enter this territory by showing how the control of the selection of the alternative transformations based on the desired usability criteria can be an appealing way to ensure the usability of the generated artifact.**

Keywords—User Interface; Usability Driven Model Transformation; Parameterized Transformation; Design Decision Control

I. INTRODUCTION

In recent years, the software development is moving towards Model Driven Development (MDD) process. It provides an automatic process that builds the software system based on the construction and maintenance of models at several abstraction levels to drive the development process [1]. Within this context, the Model Driven Engineering (MDE) initiative has attracted the interest of the Human Computer Interaction (HCI) community. It has gained a wide acceptance as an appropriate approach for the development of user interface which are able to adapt their layout with respect to the context of use wherein the interaction takes place [2]. Calvary et al. [3] used the term *Multi-target* to indicate this kind of user interface.

Usually, the MDE development process transforms a platform-independent model (PIM) into one or more platform-specific models (PSM), which are transformed into code (code model - CM) [1]. So, a model transformations process basically converts one model (source model) to another (target model). There are several ways to transform a source model into a several target models which may be equivalent from the functional perspective but may differ in their usability. Therefore, there is a need to identify those transformations that produce models with the desired usability attributes.

To address this issue, this paper presents an approach for driven the model transformations process by usability criteria. The selection of the appropriate alternatives can greatly differ depending on the usability criteria that are chosen.

The proposed approach is composed of two stages: the rule definition stage and the transformation stage. During the first stage, the designer establish the relation between the alternatives transformation rules and the usability criterion which are able to meet. In the second stage, the model transformations are executed to ensure the usability of the user interface artifact. The aim of the proposed approach is to provide a practical support to improve current model transformations practices targeting the user interface adaptation to the context of use where usability issues are neglecting.

We structure the remainder of this paper as follows. Firstly, a brief look to the related works is presented in section 2. Next, section 3 details the proposed approach. Section 4 shows a case study illustrating the application of the proposed approach. Finally, the proposed approach is briefly outlined and future perspectives are given.

II. RELATED WORK

This section presents an overview of the most cited research studies in the literature that deal with usability in an MDE approach. The analysis of these proposals gives us a framework to propose our contribution.

In fact, recent studies have begun to explore the problem of integrating the usability issues in an user interface development process which follow MDE principles. Some proposals have demonstrated that assessing usability early at the development process (since the conceptual model) is an appealing way to ensure the usability of the generated user interface. We quote for example the proposition of [4] and that presented in [5]. The usability is evaluated since the conceptual model to detect potential problems. Then, some changes are recommended at the design. By means of model transformation and explicit traceability between models, the performed changes directly reflected into the intermediate artifact avoiding usability problems in any future user interface obtained as part of the transformation process. Besides the lack of details about how to measure the usability attributes and to interpret their scores, these methods exclusively focus their research efforts on the evaluation and ignore the improvement issue.

The proposal of [6] is among the pioneers that address the usability issues during the adaptation process. It exploits the mapping notion to control the user interface adaptation according to explicit usability criteria. The transformation was done by *pattern* to ensure *homogeneity-consistency*. This proposition lacks of any detail about its feasibility and does not specify the relation between usability attributes and

978-1-4673-5637-4/13/$31.00

transformation rules.
Other proposals evaluate the usability of a user interface generated with an MDE approach. We quote for example proposals presented in [7], [4] and [8]. The usability evaluation is based on the system code and on the generated interface. This makes their application during the model transformations process difficult.
Considering the research works just mentioned, it becomes clear that drive the model transformation process by usability criteria is still an immature area. Therefore many more research works are needed. In order to covers this need, the present paper goes beyond the current proposals and shows how model transformations can be a suitable environment to ensure the usability of the generated artifact. By analogy to some attempts to drive the model transformation by others quality attributes as mentioned in [9] and [10], we propose to drive the model transformation process by the desired usability criteria. A set of practical usability criteria is inserted as a parameter to the transformation engine. The objective is to make the selection of the alternative transformations based on these criteria.

III. PROPOSED METHOD TO ENSURE USABILITY

In a model driven development approach, models and model transformations are the primary artifact of the development process. The mode transformations execution takes as input a model transformation definition. The model transformations definition contains transformation rules that relate constructs from the source model to constructs in the target model. When a construct from the source model have more than one possible transformation we talk about alternative transformations. We propose to select the adequate alternative transformations depending on the usability criterion which is able to meet (maximize).

A. Overview

Our proposal extends the model transformation process proposed in the Cameleon framework [3]. In fact, we opted for the extension of the Cameleon framework since it present a unifying framework for the development of multi-target user interface. The applicability of our method is shown in this paper through the Cameleon-compliant method presented in [11]. The choice of such method is motivated by two main criteria. It follows the MDE principles and uses the BPMN [12] notion to define the user interface models. The BPMN notation is built on the Petri networks, which allows the validation of the user interface models.
The Cameleon framework initiates three types of model transformations: reification, translation and abstraction. In the present paper, we focus our interest on the reification process. In particular, we concentrate on the reification step which takes as input the Abstract User Interface (AUI) model and generate the Concrete User Interface (CUI) model (AUI2CUI). We believe that our proposal can be extended to covers the others step of the reification process.
The AUI2CUI reification step associates to each Abstract Interaction Object (AIO) coming from the source model a Concrete Interaction Object (CIO) in the target model. Given an AIO, there may me several correspondent CIO allowing the same functionality but with different usability properties. For example, user interfaces of Fig. 1 are equivalent from the functional perspective. However, from the non-functional perspective they do not satisfy the same usability criteria. Solution b) allows better user guidance than solution a). It displays the measurement unit of the temperature and the range of accepted values. Hence, the *prompting* property is well addressed (satisfied) in solution b). In solution c), user is prevent to make error (i.e. typos) while entering the temperature value. Thus, the *error prevention* usability property is well fulfilled in solution c).

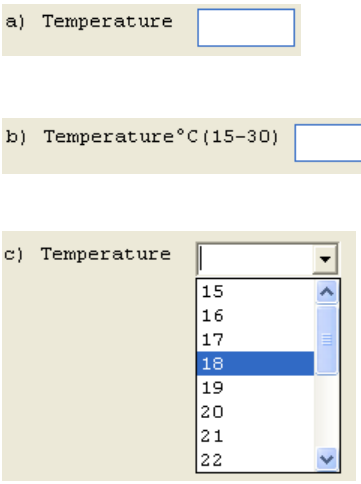

Fig. 1. Three functionally-equivalent user interface that differ from the set of usability criteria used to produce them.

It becomes clear that given a construct from the source model there may be several alternative transformations. Each one contributes to reach (maximize) a specific usability criterion. Starting from this report, we adapt the reification process of the Cameleon framework by adding a set of usability criteria on which the selection of the adequate alternative transformations will be done (see Fig. 2).

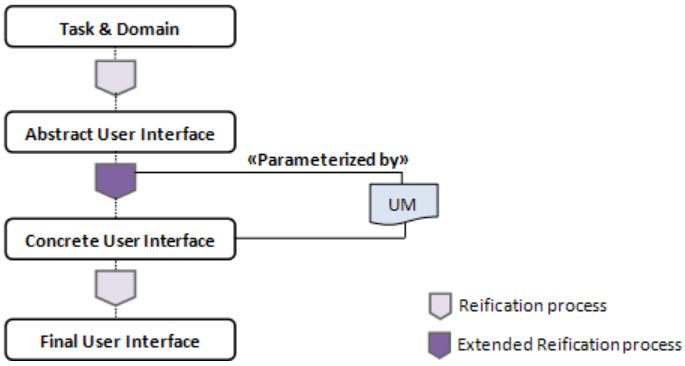

Fig. 2. The Extended Cameleon Framework.

The basic idea of our proposal is to insert a set of practical usability criteria (properties) as a parameter to the model transformations engine during the specification of the transformation rules. To do this, we reformulate the parameterized transformation principles initiated by [13]. It consists on a model transformation based on a parameter. The aim is to improve new functionalities (values, properties, operations)

or to change the application behavior (activities). For that purpose, the designer has to specify the parameter which is intended to be inserted during the transformation definition phase. In the present paper, the parameter is a usability model which contains the desired usability criteria to be satisfied (maximized) by the generated model.
Next, we detail the main activities of the proposed model transformation technology.

B. USABILITY CRITERIA SPECIFICATION

Usability is a difficult concept to quantify. It has several dimensions and several factors seem to impact upon it. Previous studies [14], [5], [15] have identified a number of factors that contribute to usability of the user interface.
In this paper, the problem we intend to address is to isolate the most important of these criteria and work out a means of characterizing the impact of each of them to the alternative transformation selection. During the extraction of the most relevant usability criteria, the relation between each of them and the context features such as the user experience or the screen size of the platform being used is kept in mind. As is already mentioned, the objective is to enhance the usability of a multi-target user interface during the development process. Moreover, we only select usability criteria which can be measured quantitatively and can be modeled in a formal way. This will allow the full automation of the development process. We note that the user experience is closely related to usability criteria that should increase the user guidance means available in the user interface. Hence, usability criteria such as *prompting* and *error prevention* are crucial to take into consideration during the user interface design.
Interactive system is a system that allow certain level of control by the human agent. Hence, the computer must process only actions requested by the users and only when requested to do so. Therefore, usability criterion such as *explicit user action* is crucial and must be considered.
The screen size of the interactive platform (device) may affect some usability criteria such as the *information density* and the *brevity*. A large screen size is generally characterized by a high information density and low brevity. Thus, it is essential to investigate the impact of such usability criteria in our case. It should be noted that although we only used few usability criteria to better explain their impact to the model transformation, our model can be extended to covers many more usability criteria in further works.
In order to formalize our proposal, we propose a usability metamodel[1] which is composed of hierarchy with two levels: subcharacteristics and attributes.

- Subcharacteristic: A set of abstract concept used to define usability.
- Attribute: An entity which can be ensured during the model transformation process.

It is compliant with the ISO/IEC 9126-1 model [16]. In fact, the ISO/IEC 9126-1 usability model deals with the characteristics of the product itself and can be used to evaluate the intermediate artifacts. Fig. 3 show our proposal metamodel.

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

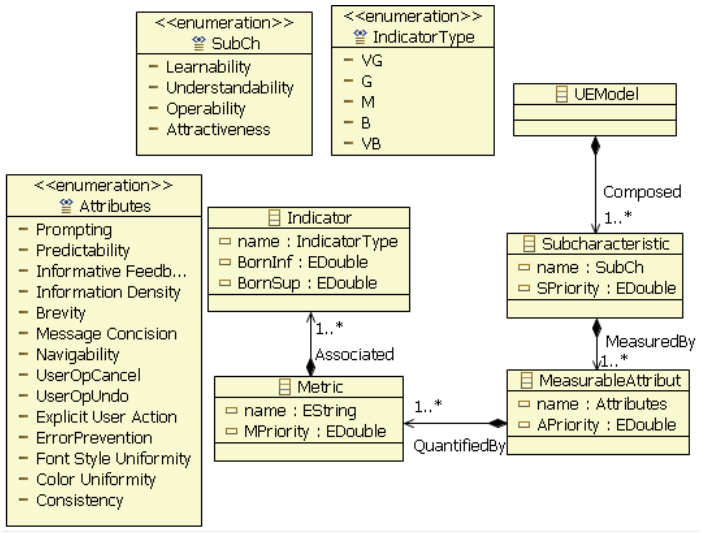

Fig. 3. The Proposed Usability metamodel.

The next section shows the impact of the specified usability criteria to the transformation alternative selection according to the underlying method.

C. USABILITY AND THE DESIGN DECISIONS CONTROL

We used Kermeta (Kernel meta-modeling) [17] as a transformation language to implement our approach. It allows the description of both structure and behavior of models.

1) Prompting.: The *Prompting* usability property refers to the means available to advise, orient, inform, instruct, and guide the users throughout their interactions with a computer. A simple example of the prompting property is illustrated by the addition of specific information to inform user about the required format while specifying data. The Listing 1.1 shows the kermeta code of our proposition to ensure the prompting property.

```
operation UIFieldTreatment(AUIModel:AbstractUserInterface,
     uic : CollapsedUIUnit,uiw : UIWindow) is do
var lnk : Link init getAllLinks(AUImodel).detect{c|c.
     uicomponent.name == uic.name}
var componentnature : Nature init uic.nature
if (componentnature == Nature.Specify) then
     createFieldIn(uiw,uic,lnk)
     if (PromptingSupport(paramModel) then
          createStaticField(uiw,uic,lnk.name,lnk.
               uicomponentannot.dataformat)
     else
          createStaticField(uiw,uic,lnk.name,"")
     end
end
```

Listing 1. The *Prompting* implementation

Having restored the annotation attached to the abstract component through the link *lnk*, all information about the component (name, nature, required format, etc.) is available. If the input abstract component has the nature *Specify*, the program associates with this component and edit field and a label in the concrete user interface model. If the prompting property is required in the target model, the program adds the specific information (the required data format in the example) to the label.

2) Error Prevention.: The *Error Prevention* property refers to the available means to prevent data entry errors. We propose to create a dropdown list (or radio button) instead of an edit field when the data to be inserted has a set of possible values.

```
if (ErrorPreventionSupport(paramModel) then
    if (lnk.uicomponentannot.conceptNB > 5) then
    createDropDownList(uiw,uic,lnk)
    else
        from var i : Integer init 0 until i == conceptNB
             loop
        createRadioButton(uiw,uic,lnk)
        end
    end
end
```

Listing 2. The *Error Prevention* implementation

The number of the manipulated concept is the main factor that affects the choice of the target element. If the number of manipulated concepts is greater than a threshold (5 in the example) the input element will be realized by a dropdown list. Otherwise it will be realized by a set of radio button.

3) Information Density.: The *Information Density* refers to the degree in which information will be display to the user in each interface. User interface should not be too dense. Information density can be measured with respect to the total number of interface elements which should not exceed a threshold. Having a total number that exceed the threshold, we propose to associate a window to each unit suite[2] which is usually realized by a panel. In the kermeta code of the Listing 1.3, we used an example of 20 elements as a threshold [5].

```
 var Totalelement : Integer init Integer.new
 Totalelement := NBelemnt(AUImodel)
 if (SupportInformationDensity(paramModel)) then
    if (Totalelement>20) then
       uig.uiunitsuit.each{uius|
       var uiw: UIWindow init UIWindow.new
       uiw.name := uig.name
       result.uiwindow.add(uiw)
       uius.collapseduiu.each{cuiu| createUIField(
            inputModel,cuiu,uiw)}
       }
   end
end
```

Listing 3. The *Information Density* implementation

4) Explicit User Action.: The *Explicit User Action* refers to the relationship between the computer processing and the actions of the users. The computer must process only actions requested by the users and only when requested to do so. For example, each data entry (edit field, radio button, check box, dropdown list) should be ended by an explicit validation action by the user. The Listing 1.4 shows the implementation such property to the check box element.

```
operation createCheckBoxP(uip :UIPanel,uic : CollapsedUIUnit
    ,lnk : Link) is do
    var ddl : UICheckBox init UICheckBox.new
    ddl.name := lnk.uicomponentannot.data
    ddl.items:= lnk.uicomponentannot.enumValues
    uip.uifieldP.add(ddl)
    if (SupportExplicitUserAction(paramModel)) then
        createButtonP(uip,"Ok")
    end
end
```

Listing 4. The *Explicit User Action* implementation

[2]unit suite: a set of interaction elements grouped in logical groups from the interaction perspective

5) Brevity.: The *Brevity* concerns workload with respect to the number of step (keystrokes) necessary to accomplish a goal or a task. The reduction of the effort needed to perform a task can be materialized by the elimination of the navigation between windows with respect to the relationship. If the relationship is «simultaneous» both group (source and target) will be concretized by a panel in the same window. If the relationship is «sequential» the target group will be concretized by a panel in the window associated to the source group.

```
if (SupportBrevity(paramModel)) then
    var rsp : UIRelationShip init UIRelationShip.new
    rsp:= AUImodel.uispatiotemporalrelationship.detect{rs|rs
         .source == uig.name}
    if (rsp.type.equals("Simultaneous")) then
        var uiw : UIWindow init UIWindow.new
        uiw.name := "General Window"
        result.uiwindow.add(uiw)
        var srcpanel : UIPanel init UIPanel.new
        srcpanel.name := uig.name
        uiw.uipanel.add(srcpanel)
        var trgpanel : UIPanel init UIPanel.new
        trgpanel.name := rsp.target
        uiw.uipanel.add(trgpanel)
    else
        var uiw : UIWindow init UIWindow.new
        uiw.name := uig.name
        result.uiwindow.add(uiw)
        var trgpanel : UIPanel init UIPanel.new
        trgpanel.name := rsp.target
        uiw.uipanel.add(trgpanel)
    end
end
```

Listing 5. The *Brevity* implementation

D. Discussion

The examples already mentioned are intended to give a clear outlook to the impact of some usability criteria on the selection of the adequate design decisions (alternative transformations). The objective is to ensure that the generated model includes concrete component fulfilling the desired usability properties.
The last stage in our method which is the execution of the model transformations is illustrated using a case study in the next Section.

IV. CASE STUDY

This section investigates a case study in order to illustrate the feasibility of our proposal. The purpose is to allow us to learn more about the potentialities and limitations of our proposal. The research question addressed by this case study is: *Does the proposal ensures the usability of the generated user interface artifact?*
The object of the case study is a Tourist Guide System (TGS). The scenario is adapted from [18]. The mayor's office of a touristic town decides to provide visitors a tourist guide system. The system allows the visitors to choose the visit type (tourism, shopping, work, etc.).
During the visit, the TGS offers tourists several choices of visit traverses, indicate the paths to follow and provides information about the nearby points of interests. Tourists can use the system to find places (restaurant, hotel, etc.) and know the itineraries of visits.
The system will run on terminals of visitors (laptop, PDA, mobile phone, etc.). Therefore, the user interface must adapt its layout to the context of use. As example of the context of use elements we quote the computing devices being used,

the tourist language, preference, etc. It should be able also to bring a feeling of comfort and ease of use in order to increase the user satisfaction degree.

Since the tourist guide system is large, we focus our interests in the generation of the concrete user interface for the «Search itinerary» task. We suppose to have the abstract user interface from Fig 4 as a result of the transformation of the task model «Search itinerary» following the model transformation explained in details in [11]. The result of the transformation is an XML file which is in accordance with the AUI metamodel. 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. 4.

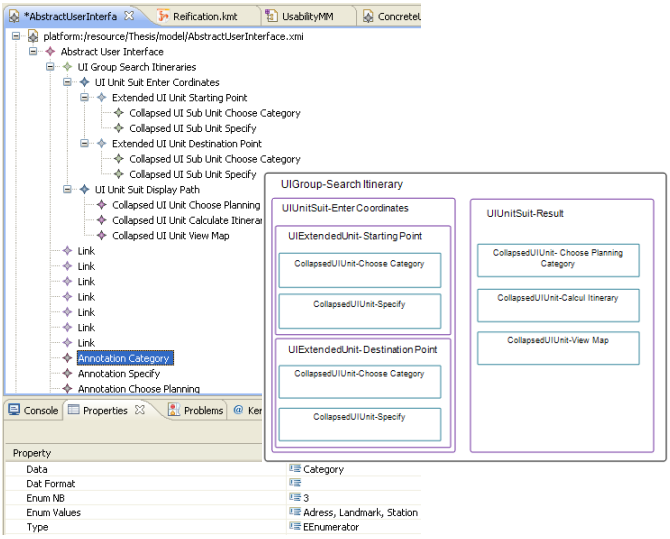

Fig. 4. Abstract User Interface.

The abstract user interface contains a *UIGroup* called «Search itinerary» which gives access to two *UIUnitSuit* called «Enter Coordinates» and «Result». The «Enter Cordinates» container gives access to specify the starting point and the destination point. The tourist should choose the category (Address, Landmark, and Station) before specifying the starting or the destination point. The validation of the coordinates allows tourist to choice the planning (Pedestrian, Cyclable, Vehicule, Metro, Train, and Bus). After that, the TGS system shows the list of possible itineraries. The TGS system can shows the list in a map.

To better explain our proposal, we start by an ordinary transformation which takes as input the abstract user interface model. This transformation allows the creation of the concrete user interface model which illustrated in Fig. 5. It should 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 (medium screen size), an Englishman as a tourist with a low level of experience. After evaluating the concrete user interface and detected potential problems, we execute a second transformation parameterized by desired usability criteria.

Although the generated concrete user interface fulfilled their objectives, it does not satisfy some usability properties. Usability properties such as *Information Density* and *Error Prevention* are not supported. So, the second transformation

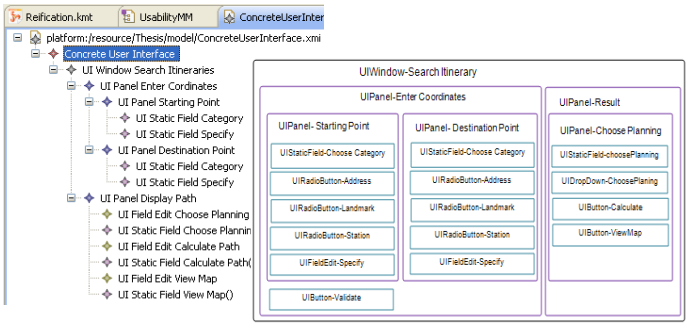

Fig. 5. Concrete User Interface (Large screen size).

will be parameterized by a usability model that conveys these criteria.

As already mentioned, the take into account of the *Error Prevention* is materialized by the association of *UIDropDownList* with the «Choose Category» *CollapsedUIUnit* instead of *UIRadioButton*. This remodeling of the user interface model reduces the total number of concrete components in the concrete model. Hence, both *Error Prevention* and *Information Density* are increased by this remodeling.

The generated concrete model is shown in Fig. 5.

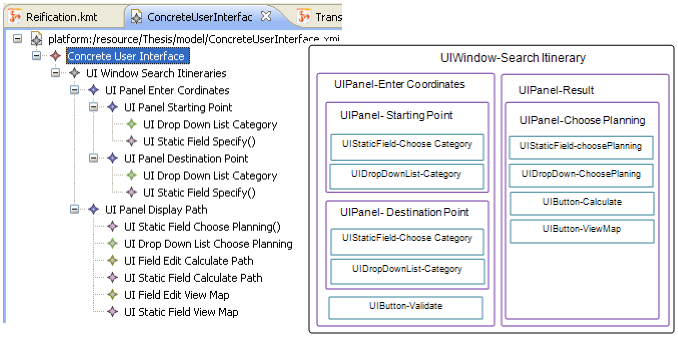

Fig. 6. Usable Concrete User Interface for large screen size.

The migration to the PDA platform «iPAQ Hx2490 Pocket PC» raises a new redistribution of the user interface elements. The small screen size (240x320) is not sufficient to display all information. The *Information Density* is the main usability property that must be taken into account in this case. 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»). Fig. 7 shows the deduced concrete user interface for the underlying platform.

We note that the take into account of the *Information Density* property has influenced negatively the *Brevity* property. This raises a new issue about the contradictory effect of the usability properties. In addition, thresholds considered to select between alternative transformations are also influenced by the screen size. For example, the threshold used for the *Information Density* is influenced by the screen size of the platform being used and the maximum number of manipulated concept. Therefore, it is recommended to conduct many more experiments in order to produce a repository that relate each usability property with the context feature which may influence it.

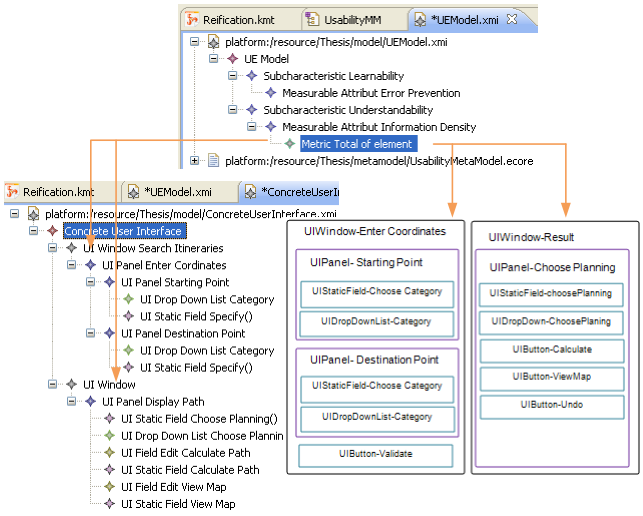

Fig. 7. Usable Concrete User Interface for small screen size.

Learned Lesson.

The case study has been useful in that it has allowed us to learn more about the potentialities and limitations of our proposal and how it can be improved.

The usability driven model transformation process may be an appealing way to enhance the usability during the transformation process. It is highlighted that the selection of alternative transformations based on the usability criteria they are able to meet is an appealing way to ensure that the generated artifact contains components that fulfil (maximize) these criteria.

The aim of this paper is to show the feasibility of applying our proposal in an MDE approach. The accuracy is the main question to solve in further works. The thresholds are extracted from existing studies that does not take into account the context variation. For example, the information density indicators are strongly influenced by the screen size. Therefore, many more experimentations are needed in order to propose a repository of thresholds in several cases (medium screen size, small screen size, large screen size). The same things for other metrics which are influenced by the context variation.

Another important aspect which should be solved is the contradictory affect of usability attributes. For example, for computing platform with small screen size the information density and the brevity has a contradictor affect. Increasing the information density will decrease certainly the brevity attribute. Therefore, many more experiments are needed to provides a catalog which can guide the analyst about the most relevant usability attributes to consider during the usability specification step.

Finally, the case study was very useful since it allows highlighting the benefits and limitations of our proposals. It illustrate the feasibility of our proposal; We can state that the method presented in this paper can be a building block of a model transformation technology where the usability criteria are taken into account during an adaptation process of a user interface to the context of use wherein the interaction takes place.

We believe that our contribution can be a building block to provide a practical support for current model transformations technologies targeting the user interface adaptation to the context of use where usability issues are neglecting.

V. CONCLUSIONS AND FUTURE WORK

In this paper, an extension of the Cameleon reification process is presented. The main motivation of the extension is to keep in mind usability issues during the transformation process. The objective is to ensure that the generated user interface fulfill the desired usability properties. To reach this objective, we build our proposal on the parameterized transformation technique. In such transformation, a parameter model is required to communicate the usability requirement. The specification of the transformation rules is made up following the desired usability properties. Consequently, the design decisions are controlled by the desired usability properties. The selection of the adequate alternative transformations depends on the usability criteria we want to maximize in the generated artifact. The case study presented is useful since it highlight the benefits and limitations of our proposal. We argue that our proposal provides a practical support to existing tendency addressing usability treatment during the development process. The usability driven model transformation concept initiated in the present paper is the main advantage of our proposal if comparing with existing one. The model transformations definition is accompanied by a proper level of detail. The execution of the model transformations is illustrated through a practical case study. However, this cannot hide the limitations of our proposals which are generally raised during the execution of the case study. We can note mainly the selection of multiple usability attributes which are not compatible and have a contradictory effect.

Several research studies can be considered as a continuation of this work. As an example, further research works are intended to perform an automatic evaluation process of the intermediate artifacts in order to detect potential usability problems. To do that, we have to propose a consolidated usability model gathering all properties which can influence the usability of a user interface. Usability attributes are intended to be evaluated at the intermediate artifact. Thus, only quantitative measures are needed. This can help in the formalization of the usability model in order to automatically evaluate the intermediate artifact. The interrelation between usability properties and the contradictory effects of properties can also be targets of attack for future work. We have to think about usability properties which can be inserted in the others level (from Task & Domain to AUI and from CUI to FUI) of the Cameleon framework.

REFERENCES

[1] D. C. Schmidt, “Model-driven engineering,” *IEEE Computer*, vol. 39, no. 2, February 2006. [Online]. Available: http://www.truststc.org/pubs/30.html

[2] J. Coutaz, “User interface plasticity: Model driven engineering to the limit!” in *ACM, Engineering Interactive Computing Systems (EICS 2010) International Conference. Keynote paper.* ACM publ., 2010, pp. 1–8, keynote paper.

[3] G. Calvary, J. Coutaz, and D. Thevenin, “A unifying reference framework for the development of plastic user interfaces,” in *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, ser. EHCI ’01. London, UK, UK: Springer-Verlag, 2001, pp. 173–192. [Online]. Available: http://dl.acm.org/citation.cfm?id=645350.650727

[4] S. Abrahão, E. Iborra, and J. Vanderdonckt, "Usability evaluation of user interfaces generated with a model-driven architecture tool," in *Maturing Usability*, 2008, pp. 3–32.

[5] J. I. Panach, N. Condori-Fernández, T. E. J. Vos, N. Aquino, and F. Valverde, "Early usability measurement in model-driven development: Definition and empirical evaluation," *International Journal of Software Engineering and Knowledge Engineering*, vol. 21, no. 3, pp. 339–365, 2011.

[6] J.-S. Sottet, G. Calvary, J. Coutaz, and J.-M. Favre, "Engineering interactive systems," J. Gulliksen, M. B. Harning, P. Palanque, G. C. Veer, and J. Wesson, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, ch. A Model-Driven Engineering Approach for the Usability of Plastic User Interfaces, pp. 140–157.

[7] N. Aquino, J. Vanderdonckt, N. Condori-Fernández, O. Dieste, and O. Pastor, "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*. New York, NY, USA: ACM, 2010, pp. 30:1–30:10. [Online]. Available: http://doi.acm.org/10.1145/1852786.1852826

[8] A. Fernandez, E. Insfran, and S. Abrahão, "Integrating a usability model into model-driven web development processes," in *Proceedings of the 10th International Conference on Web Information Systems Engineering*, ser. WISE '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 497–510.

[9] M. Matinlassi, "Quality-driven software architecture model transformation," in *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*, ser. WICSA '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 199–200. [Online]. Available: http://dx.doi.org/10.1109/WICSA.2005.56

[10] E. Insfran, J. Ángel Carsí, S. Abrahão, M. Genero, I. Ramos, and M. Piattini, "Towards quality-driven model transformations: A replication study," 2008.

[11] W. Bouchelligua, A. Mahfoudhi, N. Mezhoudi, O. Dâassi, and M. Abed, "User interfaces modelling of workflow information systems," in *EOMAS*, 2010, pp. 143–163.

[12] "Bpmn: Business process modeling notation version 1.0. (2004). availabe: http://www.bpmn.org." [Online]. Available: http://www.bpmn.org

[13] S. Vale and S. Hammoudi, "Context-aware model driven development by parameterized transformation," in *Proceedings of the First International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) held in conjunction with the CAiSE'08 Conference*. Springer-Verlag, 2008, pp. 121–133.

[14] S. Abrahao and E. Insfran, "Early usability evaluation in model driven architecture environments," in *Proceedings of the Sixth International Conference on Quality Software*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 287–294. [Online]. Available: http://dl.acm.org/citation.cfm?id=1190618.1191343

[15] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, "Usability measurement and metrics: A consolidated model," *Software Quality Control*, vol. 14, pp. 159–178, June 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1132324.1132342

[16] ISO/IEC, *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001.

[17] "Kermeta, kernel meta-modeling framework. available : http://www.kermeta.org." [Online]. Available: http://www.kermeta.org

[18] M. Hariri, *Contribution à une méthode de conception et génération d'interface homme-machine plastique*, 2008. [Online]. Available: http://books.google.tn/books?id=OKhpXwAACAAJ