An MDE Approach for User Interface Adaptation to the Context of Use

Wided Bouchelligua[1,2], Adel Mahfoudhi[2], Lassaad Benammar[2], Sirine Rebai[2], and Mourad Abed[1]

[1] University of Valenciennes
LAMIH, UMR CNRS 8530, BP: 311–59304, Valenciennes cedex 9, France
{wided.bouchelligua,mourad.abed}@univ-valenciennes.fr
[2] University of Sfax
ENIS, CES, Soukra Road km 3.5, B.P: w 3038 Sfax, Tunisia
adel.mahfoudhi@fss.rnu.tn,
{lassaad.benammar,sirine_rebai}@hotmail.com

Abstract. With the advent of new media and the delivery of recent means of communication, associated with the progress of networks, the circumstances of software use, as well as the skills and the preferences of the users who exploit them, are constantly varying. The adaptation of the User Interface (UI) has become a necessity due to the variety of the contexts of use. In this paper, we propose an approach based on models for the generation of adaptive UI. To reach this objective, we have made use of parameterized transformation principle in the framework of the Model Driven Engineering (MDE) for the transformation of the abstract interface into a concrete interface. The parameter of this transformation plays the role of the context of use. The paper develops two parts: meta-models for every constituent of the context of use and the adaptation rules.

Keywords: User Interface, Adaptation, Context of use, Model Driven Engineering, adaptation rules.

1 Introduction

The technological innovations and the evolution of the means of communication have opened new perspectives to guide the use of the usual applications. Besides, the circumstances of software use have constantly varied following the example of the skills and the preferences of the users who exploit them. This is due to the appearance of new media and the delivery of recent means of communication, associated with the progress of networks. It is not only resources of interaction that can appear and disappear, but also the objectives of the user. The latter is considered as a motive, evolving in a varied environment, according to his needs, to diverse platforms of interaction. That is why, in 1999 Thevenin brought a new concept: the plasticity of interfaces [21]. The plasticity is defined as the capacity of a user interface to adapt itself to the context of use which is denoted by the triplet <user, platform, environment>, while preserving usability.

Several approaches are proposed to make User Interfaces (UI) adaptable to the context of use. According to [18], these approaches are classified into four categories: 1) Translation Interface, 2) Reverse-engineering and migration Interfaces 3) approaches based on the markup languages and 4) model-based approach. The latter is adopted in this work because it has the advantage of applying the adaptation to the context of use of the models, leading to a strong abstraction.

The proposed approach in this paper assures the adaptation of the UI to the context of use. It builds on the concept of transformation parameterized by the context as defined within the framework of the Model Driven Engineering (MDE) [1] [8]. MDE goes beyond the framework of Model Driven Architecture (MDA) [15], which can be summarised in the elaboration of the Platform Independent Models (PIM) and in their transformation into Platform Specific Models (PSM) [1], to cover the methodological aspects. We apply the parameter setting at the level of the transformation of an Abstract User Interface (AUI) into a Concrete User Interface (CUI), whose generation is made on three phases. The first transformation parameterized by the model of adaptation describing the user, gives rise to a first CUI, which in turn is going to feed the second module of transformation. But, the latter will be parameterized by the characteristics of the platform to generate a Concrete User Interface in agreement with the preferences of the user and the properties of the interaction platform. In the last phase, the process of adaptation connected with the environmental context is launched to finish with a plastic Concrete User interface to conform to three dimensions of the context of use.

The remainder of this paper is structured as follows. Section 2 presents a state of the art on the model-based approaches for the adaptation of the UI. Next, section 3 clarifies the concept of the parameterized transformation in the MDE approach. Then section 4 describes the proposed approach in terms of context meta-models and adaptation rules. Section 5 provides a case study illustrating the approach. Finally, section 6 draws the conclusion and provides perspectives to future research.

2 State of the Art

This section is limited to the presentation of model-based approaches for UI adaptation. In fact, the Cameleon reference framework [5] represents an excellent framework of UI adaptation as it defines four essential stages for the development of the user interfaces in a pervasive environment (Fig. 1): tasks and concepts, abstract interface, concrete interface, and final interface.

In this area of research, we can quote the TERESA method [17] that supplies the tasks as a single model, and allows the generation of several interfaces for various platforms. We also cite the Comets (COntext sensitive Multi-target widgETS) [4], which proposes essentially a model for the plastic interactors that can be adapted to the variation of the screen size. Likewise, the UsiXML (User Interface eXtensible Markup Language) [23] [14] approach represents a UI approach of engineering defined according to the Cameleon reference framework. Such an approach describes a context model consisted of three constituents: user, environment and platform. But practically only the variant platform is considered during the UI generation.

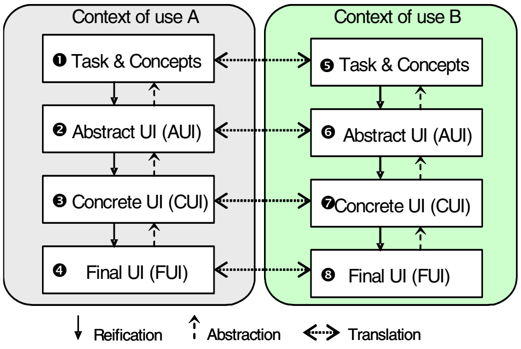


Fig. 1. Cameleon Reference Framework [23]

Sottet [19] [20] is considered as one of the pioneers to have joined his work to the Model Driven Engineering and the domain of Human Computer Interaction (HCI). His approach makes it possible to show that the concepts of the MDE could be successfully applied to the UI engineering. Sottet [19] proposes meta-models and models transformations to generate adaptable UI, and defines a general context meta-model. Based on the same approach (MDE), Hachani [11] suggests the introduction of the context of use at the tasks level rather than at the interactors level. This approach is distinguished by the definition of the generic rules appropriate to all the contexts of use. However, both approaches lack a detailed description of each constituent of the context of use. As in [20] and [11], we opt for the proposition of a model-based approach and its transformation according to the characteristics of the context. Yet, we seek to detail the context in accordance with three generic meta-models (user meta-model, platform meta-model and environment meta-model).

3 MDE Parameterized Transformation

Our objective is to handle the adaptation of the UI to the context of use (platform, environment, user). To do so,our work builds on the parameterized transformations defined by [22]. Vale [22] describes a parameterized transformation within the framework of the model driven engineering for a contextual development. The methodology proposed by [22] (left of Fig. 2) consists in defining the correspondences "match" between the model of the context and the PIM to define a CPIM (Contextual PIM). Then, an ordinary MDE transformation is used to define the CPSM (Contextual PSM).

The correspondences are assured by a parameter setting of the transformation, whose basic principle is to take into consideration the properties of the context during the specification of transformation rules (right of Fig. 2). *"A parameter specifies how arguments are passed into or out of an invocation of a behavioural feature like an operation. The type and multiplicity of a parameter restrict what values can be passed, how many, and whether the values are ordered"* [22].

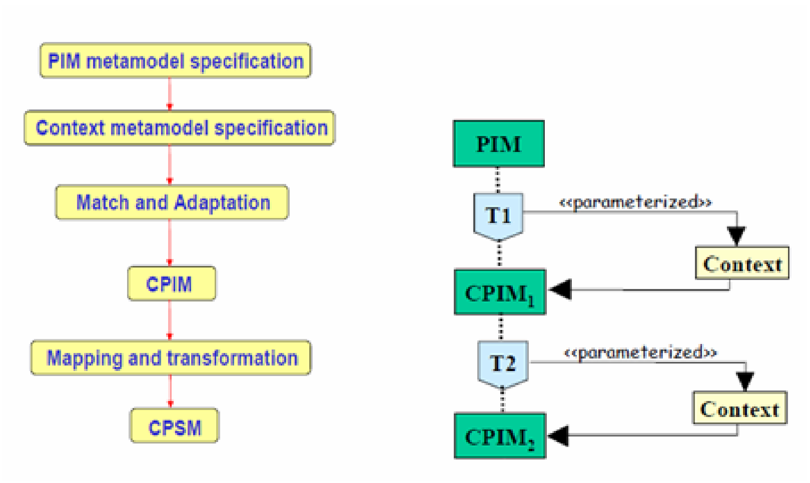


Fig. 2. Parameterized transformation [22]

The use of the parameterized transformations is envisaged with the aim of either improving new features (values, properties, operations) or changing the behaviour of an application. For that purpose, the designer has to specify the parameters which are intended to be inserted during the phase of transformation. In his work, Vale [22] proposes that these parameters are the contextual data, thus and after the transformation, the application will be in harmony with this information passed in parameter.

In the following section, we formulate the principle invoked by Vale, in favour of our approach to implement the notion of plasticity.

4 Proposed Approach

The proposed approach consists in generating the user interface automatically, using the parameterized transformation technique of the Model Driven Engineering domain

4.1 Overview of the Approach

The proposed approach in this paper is shown in Fig. 3. The abstraction levels of the Cameleon framework [5] incorporated in our approach are: abstract user interface and concrete user interface.

In our approach, the Abstract User Interface allows the transition of the specification in the modelling of the abstract components of the interface. In order to describe the Abstract User Interface and the Concrete User interface, we have resorted to the static model of interactions [3]. Aiming at applying a model-to-model transformation, we have refined the static model of the interactions of [3] in the form of two meta-models: the AUI and CUI meta-models. Indeed, AUI meta-model describes the hierarchy of the abstract components corresponding to the logical groups of interactions.

The AUI meta-model and transformations rules to obtain the AUI from the task model are detailed in [2].

The Concrete User Interface is deduced from the Abstract User Interface to describe the interface in terms of graphic containers, interactors and navigation objects. We have extended the CUI meta-model presented in [2] [16] to add the vocal components and to associate with every container of the interface a "PersonnalizationService" component containing properties used to specify the presentation of such an object as well as any object being a part of this container. Quoting for example the service "useoflanguage" which can be active if the user prefers a language other than French. If this service is activated, the attribute language allows the specification of the sought language.

The passage to the concrete level has for objective the generation of a plastic interface adapted to the planned context. Our approach facilitates the adaptation of the UI to the user, because the latter is in the centre of all the problem of the UI and everything revolves around him.

The first transformation (T1 in Fig. 3) allows the generation of the first concrete user interface (CUI1 in Fig. 3) adapted to the preferences of the user having received the information suitable to him and echoing them on this intermediate interface.

On the other side of the coin, we are interested in the injection of the characteristics of the platform used to assure the plasticity towards this context. Indeed, we opted for choosing this injection order of the characteristics for multiple reasons. On the one hand, it is around the user that revolves everything and it is his characteristics that are going to impose the choice of the platform. Besides, it is the user who decides about the device on which he even wishes to post the information. Indeed, this variation is going to require the appearance and the disappearance of the other devices of interaction. Furthermore, it is according to his preferences that the modality: graphic, hearing or even olfactive is going to be chosen. Then, in case of change at the level of one of the contextual dimensions, an adaptation is launched to protect the usability [21]. Certainly, the specific properties and the capacity characteristics of the target device have to satisfy the needs of the user. This second transformation (T2 in Fig. 3) adapts the first CUI1 to the characteristics of the platform which is going to welcome the application, from which the second CUI (CUI2 in Fig. 3) results.

So, having fixed and adapted the characteristics of the target platform to his own motivations and intentions, the user has now nothing but to choose the environment which is going to welcome the application. In fact, this environmental variant has to be in agreement with the characteristics of the user and the target platform. It is the profile of the user, defined as being a first entity for the process of adaptation, as well as these accompanied intentions, naturally, symptomatic of the platform that are going to determine the environmental aspects. The latter are going to be implemented during the process of adaptation to succeed in the generation of a plastic UI while taking into account three speeds of the context. Hence, in the third place, we are going to inject

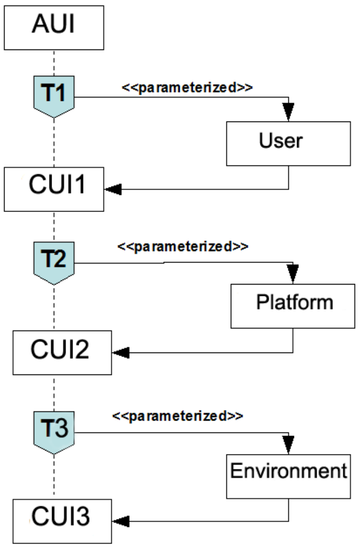


Fig. 3. Parameterized transformation for the UI plasticity

the environmental properties in the third transformation (T3 in Fig. 3) to have the interface (CUI3 in Fig. 3).

Therefore, the generation of the concrete interface (CUI) is made on three phases. To do so, we have to establish three meta-models (the user, platform and environment meta-models), so as to implement the transformation principle to illustrate the process of adaptation.

4.2 Context of Use Meta-models

The context is identified by many teams [5] [18] [23] [20] [12] by the triplet <User Platform, Environment>. Thus, three categories of contextual information can be distinguished [7]:

– So much information pertaining to the platform (processor, memory, peripheral equipments, connection network, the size of the display screen, and the available interaction tools …)
– Those relative to the user (his profile, his current activity, his preferences, his habits, his cultural characteristics…)
– The information corresponding to the environment (light, noise, geographical localization …)

4.2.1 User Meta-model

The user model has to contain information allowing the characterization of the user. Our meta-model (Fig. 4) builds strongly on the work of [10] and [6]. Such information contained is classified in four categories:

- Information staff (the name and the first name of the user, the age, the kind)
- Knowledge (The expertise level of the user in computer science, the expertise level regarding task or manipulated concept)
- Preference (The modality of interaction (graphic, vocal, olfactive, tactile, etc.), police, the character size, colour and the sound volume)
- Capacity (physical (sensory and engine) and cognitive capacities).

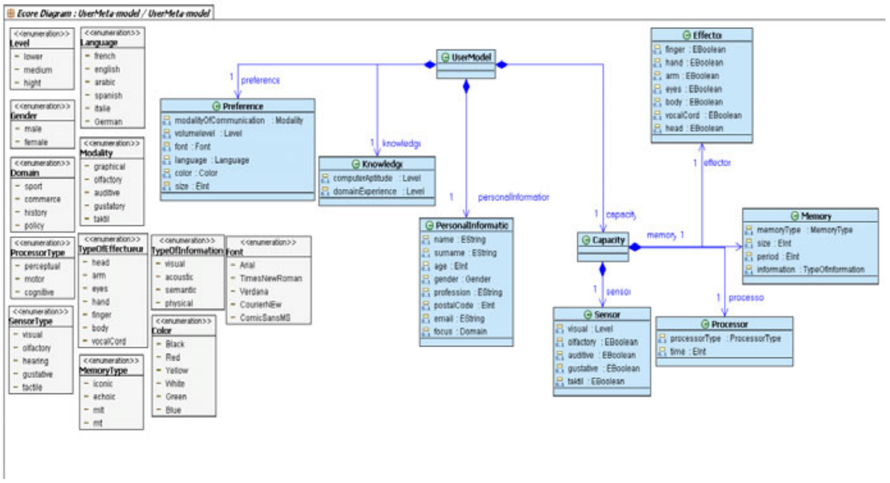

Ecore Diagram : UserMeta-model / UserMeta-model
<<enumeration>> Level - lower - medium - hight
<<enumeration>> Language - french - english - arabic - spanish - italie - German
<<enumeration>> Gender - male - female
<<enumeration>> Modality - graphical - olfactory - auditive - gustatory - taktil
<<enumeration>> Domain - sport - commerce - history - policy
<<enumeration>> ProcessorType - perceptual - motor - cognitive
<<enumeration>> TypeOfEffectueur - head - arm - eyes - hand - finger - body - vocalCord
<<enumeration>> TypeOfInformation - visual - acoustic - semantic - physical
<<enumeration>> Font - Arial - TimesNewRoman - Verdana - CourierNew - ComicSansMS
<<enumeration>> SensorType - visual - olfactory - hearing - gustative - tactile
<<enumeration>> Color - Black - Red - Yellow - White - Green - Blue
<<enumeration>> MemoryType - iconic - echoic - mlt - mt
UserModel
Preference: modalityOfCommunication : Modality; volumelevel : Level; font : Font; language : Language; color : Color; size : EInt
Knowledge: computerAptitude : Level; domainExperience : Level
PersonalInformatic: name : EString; surname : EString; age : EInt; gender : Gender; profession : EString; postalCode : EInt; email : EString; focus : Domain
Capacity
Effector: finger : EBoolean; hand : EBoolean; arm : EBoolean; eyes : EBoolean; body : EBoolean; vocalCord : EBoolean; head : EBoolean
Memory: memoryType : MemoryType; size : EInt; period : EInt; information : TypeOfInformation
Sensor: visual : Level; olfactory : EBoolean; auditive : EBoolean; gustative : EBoolean; taktil : EBoolean
Processor: processorType : ProcessorType; time : EInt
1 preference
1 knowledge
1 personalInformation
1 capacity
1 effector
memory 1
1 sensor
1 processor

Fig. 4. User Meta-model

4.2.2 Platform Meta-model

Although most of the work done on plastic UI made adaptation to the platform, it does not provide a complete and detailed meta-model. The existing approaches are limited to its description at a high abstraction level or the description of the display surface of the platform which represents the most used interactional resource in the adaptations made so far. However, the adaptation can be prepared in the presence and absence of the other interaction devices. For example, if we do not have a mouse, we can suggest as a form of adaptation using a vocal interactor where the activation of the actions will be made vocally. Fig. 5 presents our platform meta-model [16] [2]. Generally, the platform consists of:

- Calculation resources represented in Fig. 5 by the "ComputationalCapacities" class. These resources does not only include the material aspect, such as the memory or processor but also the software aspect as the supported operating system;
- Interaction resources that are the input-output devices represented in our meta-model by the "InteractionDevices" class. We identify two classes of interaction devices: the input devices (InputDevice class in Fig. 5) and the output devices

(OutputDevice class in Fig. 5). Certain devices inherit both classes and are thus input/output devices, such as the touch screen.

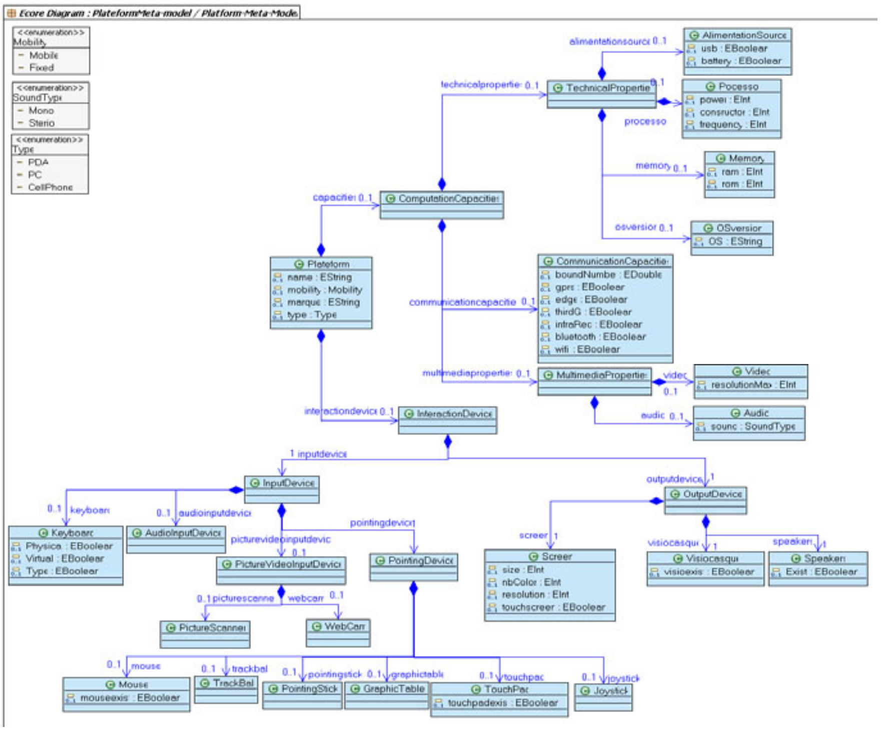


Fig. 5. Platform Meta-model [16]

4.2.3 Environment Meta-model

In this meta-model, illustrated in Fig. 6, we try to cover all the environmental facets of the context susceptible to react directly or indirectly on the interactive system. In fact, we are trying to take into account the maximum of environmental aspects. Therefore, our meta-model consists of four classes that explain the general characteristics.

- The first class characterizes the ambient environment that surrounds the interactive system "AmbientEnvironment". But with the invasion of the ubiquities computer science, the ambient conditions are changeable from one moment to another. This class inherits three under classes: "ClimaticEnvironment", "LuminousEnvironment" and "SonorousEnvironment".
- The second class composing our meta-model is the class "TemporalEnvironment".
- As for the third class, named "SocialEnvironment", it characterizes the social environment receiving the interactive system. This class is decorated with a single attribute: "atmosphere" of type "Atmosphere" enumeration.

- To specify the place receiving the application, we used the fourth class named "SpatialEnvironment". Indeed, this class gives information about the geographical location of the interactive system.

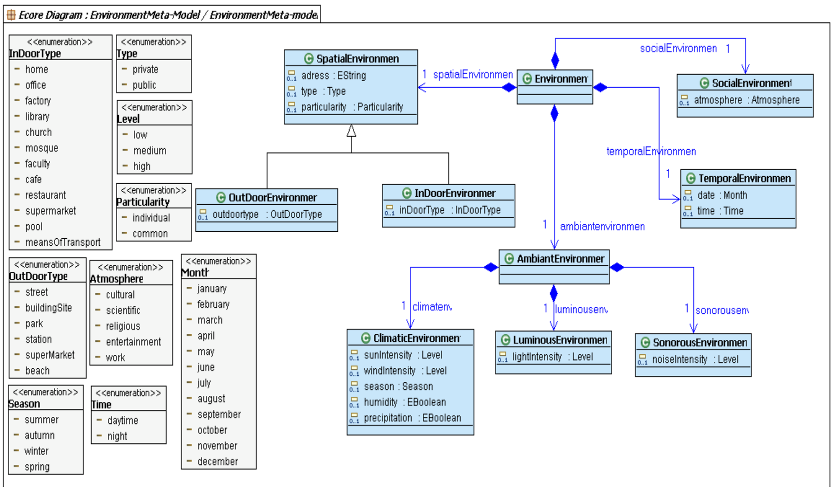


Fig. 6. Environment Meta-model

4.3 Generation Process of CUI and Adaptation Rules

The generation stages of the Concrete User Interface lean strongly on the work of [21] and [14]. The three transformations of the approach are developed with the transformation language Kermeta [13]. The transformation of an ***AUI*** into a ***CUI1*** (T1 transformation) is implemented by the following four stages:

- Creation of the application: creation of the application in the "ConcreteUserInterface" target model by the "AbstractUserInterface" of the source model;
- Realization of the abstract containers;
- Choice of the interactors;
- Definition of the navigation.

We have developed a set of rules allowing T1 transformation. As an illustration, in what follows, we clarify the stage of interactor's choice. This stage aims at associating the adequate interactor with the abstract component of ***AUI***. Such a choice depends on the properties of the abstract component: its type (Input or Output) its nature (Specify, Select, Turn …) and the user preferences.

The extract of the following code transforms every abstract component of the "CollapsedUIUnit" type into a "UIField" and appeals to the "UIFieldSpecification" method for the choice of the appropriate interactor. In that case, it is a question of executing the interactor's choice for an abstract component of the "Specify" nature.

```
operation createUIField(inputmodel :
AbstractUserInterface,collapseduiunit: CollapsedUIUnit,
uiw : UIWindow)      is do
UIFieldSpecification(inputmodel,collapseduiunit,uiw)
end
//UIFieldSpecification
operation UIFieldSpecification( inputmodel :
AbstractUserInterface,uic : CollapsedUIUnit,uiw :
UIWindow) is do
var lnk : Link
lnk := getAllLinks(inputmodel).detect{c|stdio.writeln
("link" + c.uicomponent.name)c.uicomponent.name ==
uic.name}
//Specify
var nat : Nature init uic.nature
var tp : AnnotationType init lnk.uicomponentannot.type
if (nat == Nature.Specify) then
createStaticField(uiw,uic,lnk)
createFieldIn(uiw,uic,lnk)
end //rest of code
end
```

Several existing characteristics in the model of the user can have an impact during the realization of the ***AUI***. Certain characteristics have an impact on the choice of the concrete object of interaction to know the preference of the user in terms of the modality of communication. The impact is thus expressed in terms of the reshaping of the interface. The extract of the Kermeta code below illustrates the impact of the preference modality of communication on the realization.

```
  operation       transform       (       inputModel       :
AbstractUserInterface,   paramModel   :   UserModel)   :
ConcreteUserInterface is do
    AUI2CUI := Trace <UIElement, CUIElement>.new
    AUI2CUI.create
    result := ConcreteUserInterface.new
    var        modpref        :        Modality        init
getPreference(paramModel).modalityOfCommunication
    if (modpref == Modality.graphical)  then
            stdio.writeln("Graphical Modality")
            //Graphical treatment
    else if modpref == Modality.auditive then
            stdio.writeln("Auditive Modality")
          //Auditive treatment
    end
  end
```

Other characteristics in the model of the user influence the properties of the objects of interactions rather than the choice of concrete object. The extract of the following code shows the function allowing the creation of a service (createServicePerso method). It shows the activation of the two services "useoflanguage" and "useof-tooltip" as example. The latter is activated if the user does not have strong computer capacities (computer aptitude).

```
  operation  createServicePerso(nameuiw  :String,pref  :
Preference,knl  :  Knowledge)  :  PersonnalizationService is
do
      var    srv    :    PersonnalizationService    init
PersonnalizationService.new
      srv.name :=nameuiw
      if pref.language != Language.french then
       srv.useoflanguage := true
       srv.language := pref.language.name
      else
       srv.useoflanguage := false
      end
      if knl.computerAptitude != Level.hight then
       srv.useoftooltip := true
      else
       srv.useoftooltip  := false
      end //rest of code
      result := srv
  end
```

The obtained ***CUI1*** is the source model of the second transformation that takes as parameters the characteristics of the platform. We have addressed the impact of the property screen size and inputting/outputting devices of the platform. The following code produces the testing for the required devices of graphical or vocal interaction.

```
  operation transform ( inputModel :
  ConcreteUserInterface, paramModel : Plateform)  :
  ConcreteUserInterface is do
  CUI2CUI1 := Trace <CUIElement, CUIElement>.new
  CUI2CUI1.create
  result := inputModel
  var width : Integer init
  getScreen(getOutputD(getID(paramModel))).width
  var height: Integer init
  getScreen(getOutputD(getID(paramModel))).height
  getCUIWindow(inputModel).each{uiw1|
  if (MouseExist(paramModel) and ScreenExist(paramModel)
  and    KeyboardExist(paramModel)) or
  (TouchPadExist(paramModel)and ScreenExist(paramModel)
  and KeyboardExist(paramModel)) or
  TouchscreenExist(paramModel)      then
  //rest of code
  else
  stdio.writeln("Inexistant Device")
  end}
  getVocalGroup(inputModel).each{vg|
  if VisiocasqueExist(paramModel) or
  (MicrophoneExist(paramModel)and ScreenExist(paramModel)
  and then
  getVocalForm(vg).each{vf|
  //rest of code
  else
  stdio.writeln("Inexistant Device")
  end}
  end
```

The third transformation injects the properties of the environment that will host the application. The impact of environment properties does not affect the objects of interaction, but affects the existence or nonexistence of services interface. The following code shows the activation of service "useofbackground".

```
getService(inputModel).each{srv|
if(getLuminousEnv(getAmbiantEnv(paramModel)).lightInten
sity == Level.hight) or
(getSocialEnv(paramModel).atmosphere ==
Atmosphere.religious) or
(getSpatialEnv(paramModel).getMetaClass() ==
OutDoorEnvironment and getTemporalEnv(paramModel).time
== Time.daytime) then
srv.useofbackground :=true
srv.background := BackGroundType.light
end
```

5 Illustrating Example

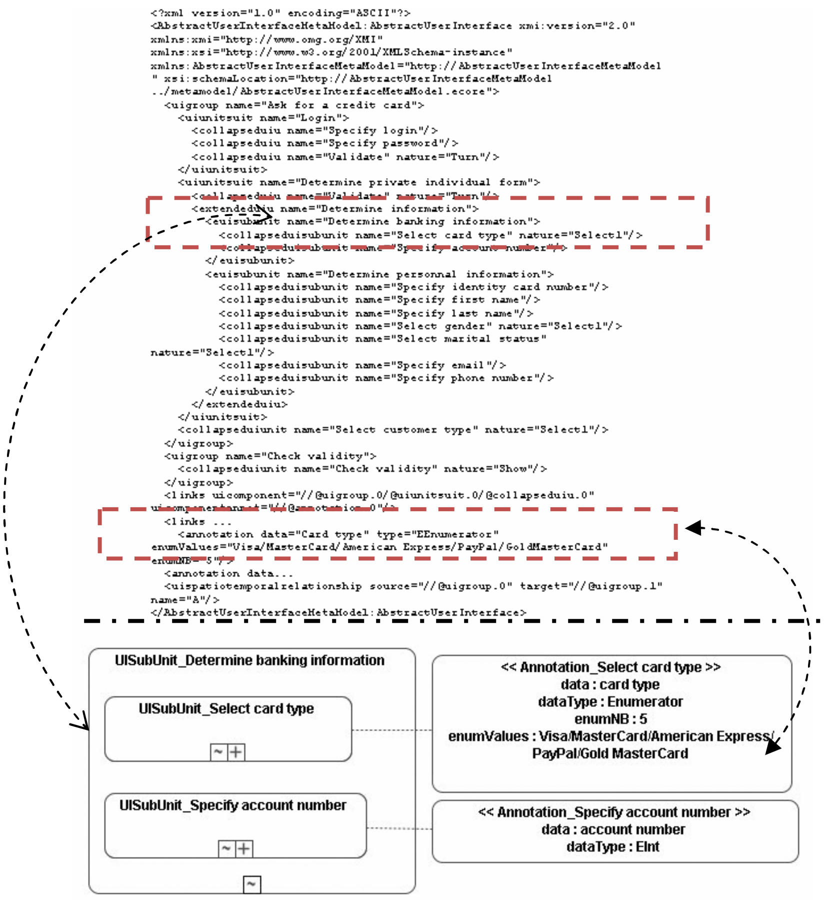
The case study relates to a credit card request by a customer. This application is adapted to the context of use. The following scenario illustrates this adaptation on a precise case. Sarra is connected to the site of the bank to launch her request of credit card. She has to log in first of all by introducing her user name and password. Then she has to choose her type (private individual or company). Then, she is asked to choose the type of card that she seeks to obtain before filling in an information form. In this case study, the following context of use is assumed:

User={Computer aptitude="lower", font="TimesNewRoman", language="english", color="Red" size="14", Modality of communication ="graphical"}, Platform={iPAQ HX2490 Pocket PC},

Environment={Alone, atmosphere="work", light intensity="low"}.

Fig. 7 shows the abstract user interface for the process of the credit card possession. This interface contains a "UIGroup" called "Ask for a credit card". This "UI-Group" gives access to two "UIUnitSuit" ("Login" and "Determine private individual form") and "CollapsedUIUnit" ("Select customer type").

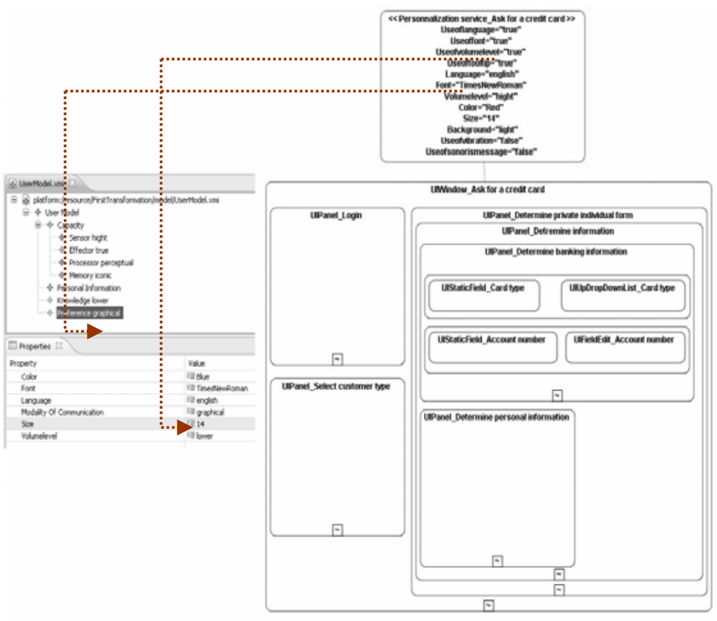
During the detection of a context change, the system is adapted. The ***AUI*** adapts itself first of all to the preferences of the user. In the following figure, we give the tree-based description of the use of our case study. In front of these characteristics, the first module of transformation uses as output the ***AUI*** model (Fig. 7) and the user model (left of Fig. 8) on which the generic transformations rules are applied, based on their respective meta-models.



```
<?xml version="1.0" encoding="ASCII"?>
<AbstractUserInterfaceMetaModel:AbstractUserInterface xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:AbstractUserInterfaceMetaModel="http://AbstractUserInterfaceMetaModel
" xsi:schemaLocation="http://AbstractUserInterfaceMetaModel
../metamodel/AbstractUserInterfaceMetaModel.ecore">
  <uigroup name="Ask for a credit card">
    <uiunitsuit name="Login">
      <collapseduiu name="Specify login"/>
      <collapseduiu name="Specify password"/>
      <collapseduiu name="Validate" nature="Turn"/>
    </uiunitsuit>
    <uiunitsuit name="Determine private individual form">
      <collapseduiu name="Validate" nature="Turn"/>
      <extendeduiu name="Determine information">
        <euisubunit name="Determine banking information">
          <collapseduisubunit name="Select card type" nature="Select1"/>
          <collapseduisubunit name="Specify account number"/>
        </euisubunit>
        <euisubunit name="Determine personnal information">
          <collapseduisubunit name="Specify identity card number"/>
          <collapseduisubunit name="Specify first name"/>
          <collapseduisubunit name="Specify last name"/>
          <collapseduisubunit name="Select gender" nature="Select1"/>
          <collapseduisubunit name="Select marital status"
nature="Select1"/>
          <collapseduisubunit name="Specify email"/>
          <collapseduisubunit name="Specify phone number"/>
        </euisubunit>
      </extendeduiu>
    </uiunitsuit>
    <collapseduiunit name="Select customer type" nature="Select1"/>
  </uigroup>
  <uigroup name="Check validity">
    <collapseduiunit name="Check validity" nature="Show"/>
  </uigroup>
  <links uicomponent="//@uigroup.0/@uiunitsuit.0/@collapseduiu.0"
uicomponentannot="//@annotation.0"/>
  <links ...
    <annotation data="Card type" type="EEnumerator"
enumValues="Visa/MasterCard/American Express/PayPal/GoldMasterCard"
enumNB="5"/>
  <annotation data...
  <uispatiotemporalrelationship source="//@uigroup.0" target="//@uigroup.1"
name="A"/>
</AbstractUserInterfaceMetaModel:AbstractUserInterface>
```

UISubUnit_Determine banking information
UISubUnit_Select card type
UISubUnit_Specify account number
<< Annotation_Select card type >>
data : card type
dataType : Enumerator
enumNB : 5
enumValues : Visa/MasterCard/American Express/
PayPal/Gold MasterCard
<< Annotation_Specify account number >>
data : account number
dataType : EInt

Fig. 7. Abstract User Interface for the process of the credit card possession (case of a private individual customer)



```
<<Personnalization service_Ask for a credit card>>
Useoflanguage="true"
Useoffont="true"
Useofvolumelevel="true"
Useoftooltip="true"
Language="english"
Font="TimesNewRoman"
Volumelevel="hight"
Color="Red"
Size="14"
Background="light"
Useofvibration="false"
Useofsonorismessage="false"
```

```
UserModel.xmi
platform:/resource/FirstTransformation/model/UserModel.xmi
  User Model
    Capacity
      Sensor hight
      Effector true
      Processor perceptual
      Memory iconic
    Personal Information
    Knowledge lower
    Preference graphical
```

Properties

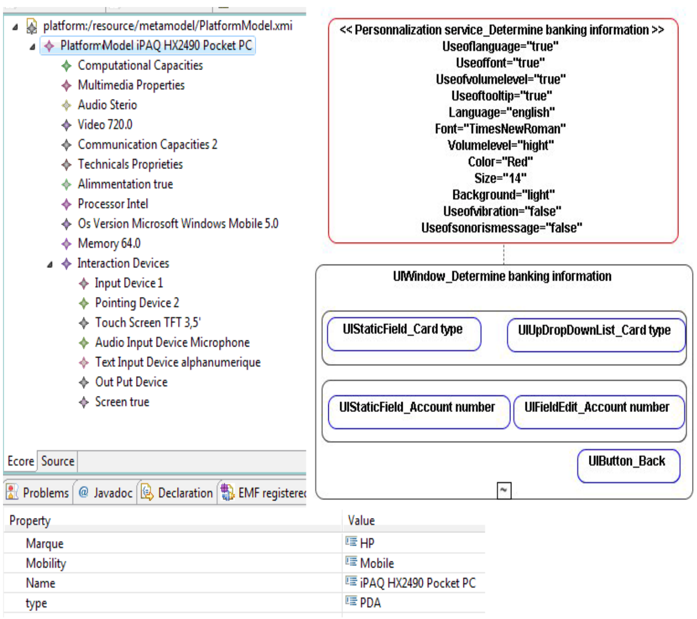
Property	Value
Color	Blue
Font	TimesNewRoman
Language	english
Modality Of Communication	graphical
Size	14
Volumelevel	lower

UIWindow_Ask for a credit card
UIPanel_Login
UIPanel_Select customer type
UIPanel_Determine private individual form
UIPanel_Detremine information
UIPanel_Determine banking information
UIStaticField_Card type
UIUpDropDownList_Card type
UIStaticField_Account number
UIFieldEdit_Account number
UIPanel_Determine personal information

Fig. 8. (Left) The tree-based description of User model. (Right) Concrete User Interface specific to the user model.

The first module of transformation consists in transforming an XML (Extensible Markup Language) file source obtained from an abstract user interface. This file is automatically generated by our AbstractUserInterface editor developed thanks to the Graphical Modeling Framework (GMF) tool [9] of Eclipse. The result of transformation is an XML file that is in harmony with the CUI meta-model. Right of Fig. 8 produces the ***CUI1*** visualized with our ConcreteUserInterface editor. The realization of the ***AUI*** is in graphic mode since the user has chosen a modality of graphic communication. A set of personalization services is activated giving as an example the service "Use of tooltip" which results from the fact that the user possesses low computing capacities.

As a concrete example, left of Fig. 9 gives the tree-based description of "iPAQ HX2490 Pocket PC" realized by EMF-based editor. The refinement of the ***CUI1*** taking into account this platform allows the generation of a concrete interface replying on the properties of this platform, as in the example of the value of the screen size (height="320" width="240"). Moreover, the choice of the appropriate interactor is related to the inputting devices that exist in the platform. In this case, we have a touch screen (TouchScreen) and a text input device (TextInputDevice). That is why the concretisation in the graphic form is possible.



```
platform:/resource/metamodel/PlatformModel.xmi
  PlatformModel iPAQ HX2490 Pocket PC
    Computational Capacities
    Multimedia Properties
    Audio Sterio
    Video 720.0
    Communication Capacities 2
    Technicals Proprieties
    Alimmentation true
    Processor Intel
    Os Version Microsoft Windows Mobile 5.0
    Memory 64.0
    Interaction Devices
      Input Device 1
      Pointing Device 2
      Touch Screen TFT 3,5'
      Audio Input Device Microphone
      Text Input Device alphanumerique
      Out Put Device
      Screen true
```

Property	Value
Marque	HP
Mobility	Mobile
Name	iPAQ HX2490 Pocket PC
type	PDA

```
<< Personnalization service_Determine banking information >>
Useoflanguage="true"
Useoffont="true"
Useofvolumelevel="true"
Useoftooltip="true"
Language="english"
Font="TimesNewRoman"
Volumelevel="hight"
Color="Red"
Size="14"
Background="light"
Useofvibration="false"
Useofsonorismessage="false"
```

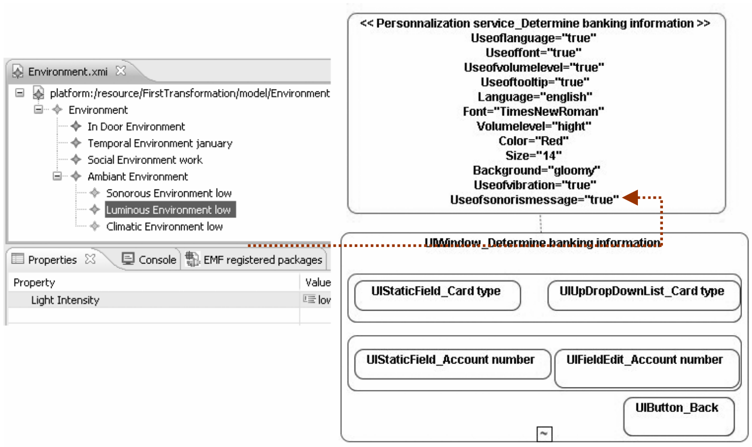
Ecore
Source
Problems
Javadoc
Declaration
EMF registerec
UIWindow_Determine banking information
UIStaticField_Card type
UIUpDropDownList_Card type
UIStaticField_Account number
UIFieldEdit_Account number
UIButton_Back

Fig. 9. (Left) The tree-based description of "iPAQ HX2490 Pocket PC". (Right) Concrete User Interface specific to the platform model.

Taking into account the properties of the platform "iPAQ hx2490 Pocket PC" (Left of Fig. 9), the transformation of ***CUI1*** (Right of Fig. 8) produces a ***CUI2*** with a re-modelling of containers. Right of Fig. 9 presents the visualization of the ***CUI2*** with our ConcreteUserInterface editor. For readability we have chosen to present only the window "Determine banking information." For the size of the screen "iPAQ hx2490 Pocket PC" and the number of manipulated concepts (>4), the realization of the abstract component "UISubUnit_Select card type" of ***AUI*** is a "UIUpDropDownList. A "UIStaticField_Card type" interactor is added, since the user does not have strong computer Capacities (computer aptitude).

Our case study is situated in a closed environment (inDoorType). As regards the ambient characteristics that specify this type of environment, it will be restored to the intensity of light as well as that of the sound level. This model (Left of Fig. 10) is going to feed the third module of transformation which will lead to the generation of a concrete interface adaptable to the context of use passing through the three elements that define it.

Taking into account the properties of environment (Left of Fig. 10), the transformation of ***CUI2*** (Right of Fig. 9) producing a ***CUI3*** with new enabled services, such as the background service whose value has become "gloomy" since the light intensity was low. Right of Fig. 10 produces the visualization of the target ***CUI3***.



Environment.xmi

platform:/resource/FirstTransformation/model/Environment
- Environment
  - In Door Environment
  - Temporal Environment january
  - Social Environment work
  - Ambiant Environment
    - Sonorous Environment low
    - Luminous Environment low
    - Climatic Environment low

Property	Value
Light Intensity	low

<< Personnalization service_Determine banking information >>
Useoflanguage="true"
Useoffont="true"
Useofvolumelevel="true"
Useoftooltip="true"
Language="english"
Font="TimesNewRoman"
Volumelevel="hight"
Color="Red"
Size="14"
Background="gloomy"
Useofvibration="true"
Useofsonorismessage="true"

Properties
Console
EMF registered packages
UIWindow_Determine banking information
UIStaticField_Card type
UIUpDropDownList_Card type
UIStaticField_Account number
UIFieldEdit_Account number
UIButton_Back

Fig. 10. (Left) The tree-based description of environment model. (Right) Concrete User Interface specific to the environment model.

6 Conclusion

In this paper, we have presented a methodology for the development of the plastic UI of an Information System. To apply "model to model" transformations, we set up two meta-models: Abstract User Interface meta-model and Concrete User Interface meta-model. The characteristic of the interface adaptation to its context of use was our primordial objective. In order to reach this objective, we proposed three meta-models describing the context of use. Encountered by a new context, a definition of a model for this context will be enough. So, our transformations rules are generic.

We foresee multiple perspectives for our work, which concern the integration of the ergonomic properties in our transformations and the determination of causality between the three components of the context of use.

References

1. Bézivin, J., Blay, M., Bouzeghoub, M., et al.: Action spécifique CNRS sur l'Ingénierie Dirigée par les Modèles. Rapport de synthèse (2005)
2. Bouchelligua, W., Mahfoudhi, A., Mezhoudi, N., Daassi, O., Abed, M.: User Interfaces Modelling of Workflow Information Systems. In: Barjis, J. (ed.) Enterprise & Organizational Modeling and Simulation. LNBIP, vol. 63. Springer, Heidelberg (2010)
3. Brossard, A., Abed, M., Kolski, C.: Context Awareness and Model Driven Engineering: A multi-level Approach for the Development of Interactive Applications in Public Transportation. In: Proceedings of 27th European Annual Conference on Human Decision-Making and Manual Control, EAM 2008, Delft, Hollande (2008)
4. Calvary, G., Coutaz, J., Dâassi, O., Balme, L., Demeure, A.: Towards a new generation of widgets for supporting software plasticity: the "comet". In: Bastide, R., Palanque, P., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 306–323. Springer, Heidelberg (2005)

5. Calvary, G., Coutaz, J., Thevenin, D., et al.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15(3), 289–308 (2003)
6. Card, S., Moran, T., Newell, A.: The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Mahwah (1983)
7. Dey, A.: Providing Architectural Support for Building Context-Aware Applications. Thèse de doctorat, Institut Technologique de Géorgie (Georgia Tech), p.170 (2000)
8. Favre, J.-M.: Toward a Basic Theory to Model: Model Driven Engineering. In: Workshop on Software Model Engineering, WISME 2004, Lisbonne, Portugal (2004)
9. GMF, Graphical Modeling Framework, `http://www.eclipse.org/gmf`
10. Habieb-Mammar, H.: EDPHA: un Environnement de Développement et de Présentation d'Hyperdocuments Adaptatifs. Thèse de doctorat, Institut National des Sciences Appliquées (INSA) de Lyon (2004)
11. Hachani, S., Dupuy-Chessa, S., Front, A.: Une approche générique pour l'adaptation dynamique des IHM au contexte. In: IHM 2009, Grenoble, France (2009)
12. Hariri, M.-A., Lepreux, S., Tabary, D., Kolski, C.: Principes et étude de cas d'adaptation d'IHM dans les SI en fonction du contexte d'interaction de l'utilisateur. Ingénierie des Systèmes d'Information (ISI). Networking and Information Systems 14, 141–162 (2009)
13. Kermeta, Kernel Meta-modeling Framework, `http://www.kermeta.org/`
14. Limbourg, Q., Vanderdonckt, J.: UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence. In: Matera, M., Comai, S. (eds.) Engineering Advanced Web Applications, pp. 325–338. Rinton Press, Paramus (2004)
15. MDA, Model Driven Architecture, `http://www.omg.org/mda`
16. Mezhoudi, N.: Méta-modèles et règles pour la plasticité des IHM, Mémoire de mastère, Institut d'Informatique et de Multimédia, Université de Gabès, Tunisie (2010)
17. Mori, G., Paternò, F., Santoro, C.: Tool Support for Designing Nomadic Applications. In: Proceedings of the International Conference on Intelligent User Interfaces, Miami, pp. 141–148 (2003)
18. Samaan, K., Tarpin-Bernard, F.: Task models and Interaction models in a Multiple User Interfaces generation process. In: Proceedings of 3rd International Workshop on TAsk MOdels and DIAgrams for user interface design TAMODIA 2004, Prague, Check Republic, pp. 137–144. ACM, New York (November 2004)
19. Sottet, J., Calvary, G., Favre, J., Coutaz, J., Demeure, A., Balme, L.: Towards Model-Driven Engineering of Plastic User Interfaces. In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 191–200. Springer, Heidelberg (2006)
20. Sottet, J.S., Calvary, G., Favre, J.M.: Mapping Model: A First Step to Ensure Usability for sustaining User Interface Plasticity. In: Proceedings of the Workshop on Model Driven Development of Advanced User Interfaces (2006)
21. Thevenin, D.: Adaptation en Interaction Homme-Machine: Le cas de la Plasticité. Thèse de doctorat, Université Joseph Fourier, Grenoble I, p. 212 (2001)
22. Vale, S., Hammoudi, S.: Context-aware Model Driven Development by Parameterized Transformation. In: Proceedings of MDISIS (2008)
23. Vanderdonckt, J.: A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 16–31. Springer, Heidelberg (2005)