User Interfaces Modelling of Workflow Information Systems

Wided Bouchelligua[1], Adel Mahfoudhi[2], Nesrine Mezhoudi[3], Olfa Daassi[3], and Mourad Abed[1]

[1] LAMIH, University of Valenciennes UMR CNRS 8530, BP: 311-59304 Valenciennes cedex 9, France
{wided.bouchelligua,mourad.abed}@univ-valenciennes.fr
[2] CES, ENIS Soukra Road km 3,5, B.P: w 3038 Sfax TUNISIA
adel.mahfoudhi@fss.rnu.tn
[3] UTIC, ESST Taha Houssein Road 5, B.P: 56, Bab Menara, 1008 Tunis TUNISIA
nesrine.mezhoudhi@yahoo.fr, olfa_daassi@yahoo.com

Abstract. In the recent years, the workflow system has increasingly gained considerable attention in the Information Systems community. It allows the integration of the various users to reach the objectives of the organization. The users operate their Information Systems through the User Interfaces (UI). Therefore there is a need to take into account the workflow model in the development process of the UI for an Information System. This paper has two objectives to attain. The first one is the proposition of a model driven approach to derive a plastic UIs of a Workflow Information System. The second objective lies in the use of Business Process Modelling Notation (BPMN) for the modelling of the interaction models (Task Model, Abstract User Interface, Concrete User Interface) with the aim of supplying more adapted models for a business expert who represents a main candidate in the success of the organization. It is through a case study applied to a real information system that our approach proves to be reliable.

Keywords: User Interface, Model Driven Approach, Workflow Information System, Plastic User interface, Business Process Modelling Notation.

1 Introduction

Nowadays, Business Process (BP) is increasingly holding a big interest in the field of Information Systems (IS). According to [9], Business Process is a structured set of activities ordered across time and space to produce a specific result for a particular customer or market. In addition, Workflow systems are defined by Workflow Management Coalition (WfMC) as the automation of a business process during which documents, information or tasks pass from one participant to another for action according to a set of procedural rules [36]. Therefore, Workflow systems help the organization to guarantee the regular adaptation of

J. Barjis (Ed.): EOMAS 2010, LNBIP 63, pp. 143–163, 2010.

its BP to its permanently evolving environment [12]. However, the adaptation of the User Interfaces (UI), through which the users process their IS, is often neglected due to the change of the organization context.

In brief, not only do interfaces have to answer the new requirements listed at the level of the workflow of an information system, but also to have the capacity to adapt themselves to the changes of their context of use. This type of interface is said to be plastic or sensitive to the context. The plasticity was defined for the first time by Thenvenin [32] as the capacity of a user interface to adapt to the context of use while preserving usability. The context of use is denoted by the triplet <user, platform, environment>. In this area of research, we can quote the TERESA method [23] which supplies a single model, that of the tasks, and allows the generation of several interfaces for various platforms. We also quote the Comets (COntext sensitive Multi-target widgETS) [8], which proposes essentially a model for the plastic interactors which are capable of adapting themselves to the context of use. To have plastic user interfaces for an IS we, firstly, opt for the integration of the workflow model as the initial model in an approach based on a set of models. Secondly, we consider the variant of the interaction platform in order to propose the adaptation rules.

The proposed approach benefits from the advantages of the domain of the Model Driven Engineering (MDE) [10]. The latter goes beyond the framework of Model Driven Architecture (MDA) [21], which can be summarised in the elaboration of the platform independent models and in their transformation into platform specific models [2], to cover the methodological aspects.

The remainder of this paper is structured as follows: section 2 gives fundamental concepts in MDE and how they can be applied in Human Computer Interface (HCI) domain. Section 3 briefly reviews the model based approaches for the development of workflow user interfaces. Section 4 describes the proposed approach to derive UIs through a case study. Finally in section 5 we draw the conclusion and provide perspectives to future research.

2 Model Driven Engineering Concepts

Since the recent adoption of the MDA by the OMG [24], the model driven approach has aroused a big interest. Then, the MDA approach has become a particular variant of the Model Driven Engineering (MDE) to cover the methodological aspects as well.

The MDE is based on three essential concepts: the models, the meta-models [31] and the transformations. These frequently used terms in the MDE and the relations between them were widely discussed in the literature [2], [3], [10], [11] and [17]. In [3] Bézivin identifies two fundamental relations: the first relation called **RepresentedBy** is connected to the notion of model, and the second called **ConformsTo** defines the notion of model with regard to that of meta-model (Fig. 1).

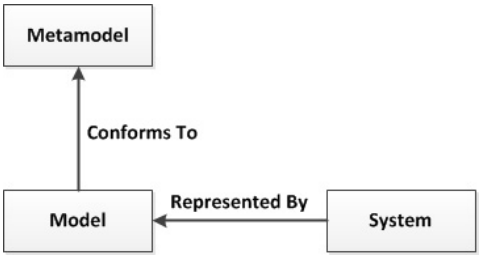

Fig. 1. Basic Notions in Model Driven Engineering

Although there are many definitions for the model concept in the literature, there is a convergence between them. Actually, they all aim at making reference to the notion of model and modelled system. Indeed, an aspect of a system is captured by a model which is linked to a meta-model in a relation called RepresentatedBy and noted μ. A meta-model is a model of a modelling language, which leads to the identification of a second relation named ConformsTo [3] [11]. Such a relation, noted by χ allows assuring the productivity of a model because it is in compliance with its meta-model. This facilitates the transformation of models. The notion of transformation is another central concept for the MDE, the mechanism of transformation allows using both Model and Meta-model notions. The power of the MDE consists in creating the transformation models, which build on meta-model corresponding to the source model and the target model. So the noted τ relation IsTransformedInto allows the automation of the transformation of a model into another.

In this section, we have chosen to present the various concepts and relations of the MDE, to show their correspondences in the field of the HCI. The purpose of our work is to benefit from MDE techniques in an approach of generating the UIs with models. The idea for the HCI models is to migrate from a conceptual to a productive aspect. The marriage of two domains turned out to be very promising, that is why so much work focused on it. The initiative of this area of research is founded by [26].

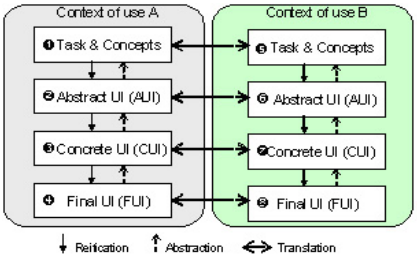

Fig. 2. Cameleon Reference Framework [35]

The work of Sottet [26] is among the first one to have joined the Model Driven Engineering and the domain of the Human Computer Interaction. His approach makes it possible to show that the concepts of the MDE could be usefully applied to the engineering of the UI. Sottet [26] proposes meta-models and transformations of models to divert plastic UI. Indeed, the Cameleon reference framework [7] defines four essential stages of user development of the user interfaces in a pervasive environment (Fig. 2): tasks and concepts, abstract interface, concrete interface, and final interface.

3 Background and Related Work

The integration of the workflow model into their approaches to derive the UI has been a recent field of research. We can, firstly, refer to the work of [22] which has demonstrated the importance to incorporate the conception of the graphic interfaces at the time of the business modelling of the application, and thus representing a first sketch of the link between the graphic interfaces and the business models. Besides, we can consult the work of [18] or even that of [29] which propose improvements to the modelling by the business process for the purpose of integrating it to the graphic interfaces. Moreover, [30] suggests an approach of development of the workflow interactive applications, which support a quick creation of initial workflow system and a derivation of the adaptable UI. Furthermore, the work of Kristiansen [19] shows the importance to use both models (task model and workflow model) in an approach of role-oriented conception of UI. She gives evidence that: "The workflow model defines what tasks need to be fulfilled and their possible ordering; hence the workflow model is suitable as a "frame" for creating task models" [19]. In the same vein of thought, Guerrero [15] confirms that "The workflow model defines what processes and tasks need to be fulfilled and their possible ordering; hence the workflow model is a "framework" for creating task model ...". In addition, [15] proposes an approach of engineering managed by the models to derive the user interfaces from the workflow model and the task model. His methodology builds on a conceptual model composed of: workflow, process, task and organizational structure. Unlike the aforementioned approaches, the recent work of Traetteberg [33] considers only one model: workflow model which is an oriented task. This model is the building stone of the conception of the dialogue model.

Based on the same fundamentals of the approaches proposed by [19] and [15], whose framework for the creation of the task model is the workflow Model, we envisage the proposition of a new approach to derive a plastic UI of a workflow information system. Besides, we opt for an easy to use notation by the domain expert who is the main candidate in the success of the enterprise.

4 Proposed Method

Aiming at illustrating the different models and meta-models, transformation rules and adaptation rules of the proposed approach, we have chosen a case

study applied to a real information system. The data-processing service company "Himilco Electronic banking" specialized in the implementation of banking solution and security of transactions and payment (e-payment) tend to conceive an application which allows the normalization of the possession process of a credit card. The scenario of the system is extracted from [1].

The workflow of this process has to satisfy a set of requirements which can be summarised in:

- Update of the rules base and the criteria of credit cards by the responsible service.
- The customer's command of a credit card from the bank web site.
- Reception of the command by the customer responsible for the check of the conformity of the data.
- Evaluation of the request by the analyst.
- The application has to satisfy all these requirements as well as those of the authentication of every system user, especially that the application is intended for banks.

In what follows, we, firstly, define a set of models and meta-models to which they are conform. Secondly, we present a description of all the transformations and we explain the major principles of UI adaptation based on a platform meta-model.

4.1 Models and Meta-Models of Method

Our approach follows the steps of the expanded Cameleon framework [16]. The extension involves the first level of abstraction (Task&Concepts). In this stage, the task model is evolved to back up the interfaces modelling and development for a workflow information system. To reach this objective, Guerrero considers a workflow model decomposed into processes that are in turn decomposed into tasks [16]. A conceptual model composed of workflow, processes, tasks and organizational units, is proposed to support this stage. This model is given in details in [16]. The workflow model describes the flow of work between the organizational units that represent the users and their business roles. Guerrero uses Petri nets [25] to describe this model. Besides, the process model shows the arrangement of the tasks from the time, space and resource perspectives (human, material and immaterial). Furthermore, the task model characterizes the division of tasks into sub-tasks connected by temporal operators from a user point of view. In [16], ConcurTaskTree (CTT) notation [23] is used to describe the task model. In short, this stage consists in the elaboration of three models: the Concept Model, the Workflow Model and the Task Models associated with the processes forming the Workflow model.

Workflow Model. Our Workflow Model is based on the Business Process Modelling Notation (BPMN) [5]. It is based on a set of simple graphic elements, easy to use and to bring together, which makes this language simple to treat by business experts. Besides, BPMN is in the course of standardization by the Object

Management Group (OMG) [24]. One of the biggest characteristics of the BPMN notation is that it is built on the Petri networks, which allows the validation of the models. By using the BPMN notation, the workflow is presented in the form of a set of activities that can be processes, sub-processes or elementary tasks. The set of activities is organized in the form of the containers which represent the partitions of the process showing a distribution of the activities by participant (an actor or a particular organizational entity). Fig. 3 shows the workflow model associated with the business process of the possession of a credit card. This business process presents one of several business processes which can be used in a bank. This workflow brings about four actors:

- The customer: a customer of the bank asking for a credit card;
- The customer responsible: the system administrator;
- The service responsible: the bank agent;
- The analyst: the financial analyst of the bank.

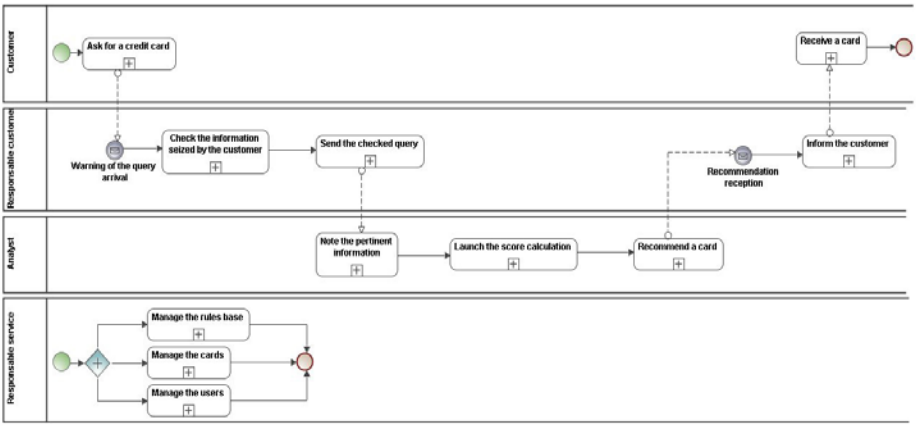

Fig. 3. Workflow Model of the possession process of the credit card

The BPMN model presents the workflow between the various actors of the system. The customer sends a request for a credit card to the customer responsible "Ask for a credit card". This request describes:

- His type (private individual, company),
- His information (Account number, ID card Number, Last name, First name ...),
- And the type of sought card.

The customer responsible verifies the information seized by the customer "Check the information seized by the customer" and sends the verified requests "Send the checked request" to the analyst. The analyst evaluates the request by noting the

relevant information "Note the pertinent information" and launching the calculation of the score "Launch the score calculation". After this evaluation, he recommends a card to the customer "Recommend a card". The customer responsible receives the recommendation and sends it to the customer "Inform the customer". The service responsible manages the rules base "Manage the rules base", the types of cards in the banking institution "Manage cards" and the users "Manage the users".

Task (Meta-) Model. According to the expanded Cameleon framework [16], each non-extended process contained in the workflow model is considered as a high-level task for a task model in the UI sense. That is why each process gives birth to a task model. The identified task models are going to feed the UI generation process proposed by the Cameleon reference framework.

Having overviewed the state of the art on the task and workflow models, we can conclude that both models share a large number of concepts. Moreover [28] proposes a methodology of alignment of a business process expressed in BPMN towards the models of user interface, more particularly, the construction of a CTT task model from the business process on the basis of a set of rules. Because these rules are associations between the elements of the BPMN business process and those of the CTT task model, reflections were drawn for the description of the task model on the basis of the BPMN notation. Our idea is confirmed by [19]: "Because of the considerable overlap in workflow and task modelling concepts, we have considered the possibility of extending BPMN so that it also can be used for task modelling". BPMN uses containers (lane) to model the process according to an organizational view. Yet, a task model is built according to a compositional view which leads to this notation extension to be able to apply the task modelling.

Our task model contains a global sub-process presented in the form of a non-extended sub-process called "CollapsedSubProcess". It should be born in mind that the BPMN notation presents a sub-process in two forms; "non-extended" (CollapsedSubProcess) or "extended" (ExtendedSubProcess). With each non-extended sub-process is associated an extended sub-process where the contents of the sub-process are detailed. This global sub-process is then decomposed in the form of abstraction levels "LevelofDiagram". A "LevelOfDiagram" presents a task decomposition level. This decomposition is made under hierarchical shape. Each level contains a set of extended sub-process "ExtendedSubProcess" spreading the set of non-extended sub-process "CollapsedSubProcess" of a superior level. Each extended sub-process "ExtendedSubProcess" contains a set of the tasks "Task" and/or non-extended sub-process "CollapsedSubProcess" as well as the relations between them. These relations are modelled by "SequenceFlow". "SequenceFlow" allows the connection between BPMN elements belonging to "ExtendedSubProcess". The entry "Gateways" allows the making of conditional connections during the execution of a process or a business sub-process. A set of events of the launching of process "StartEvent" generates a token which will be consumed by a terminal event modelled by "EndEvent". All the elements of the task model can be decorated through a constituent annotation

"Annotation". These annotations serve to decorate the tasks by the domain concepts. We have established a meta-model for the extended notation. Elements and constraints constituting this meta-model are described by means of associations and cardinalities.

The Task Model is structured in the form of levels (LevelofDiagram). It is built according to the subsequent procedure:

- **Stage 1:** Each non-extended sub-process of the workflow model is represented by a global sub-process of "CollapsedSubProcess" type;
- **Stage 2:** At level 1, specify the non-extended sub-processes and/or the elementary tasks by detailing the global sub-process;
- **Stage 3:** Use the gateways and the types of sequence flow to identify the inter activity relations (a sub-process or task) within the extended sub-process;
- **Stage 4:** Use the event types to specify the start, intermediary and end events within the extended sub-process;
- **Stage 5:** recall stages 2), 3) and 4) to identify all the sub-processes and all the elementary tasks of a sub-process to be found in the superior level.
- The end of the construction procedure of task model is characterized by the acquisition of all the actions.

Thanks to the Graphical Modeling Framework (GMF) tool [13] of Eclipse, we developed a graphic editor for our task model based on the meta-model proposed in Fig. 4. GMF is a framework that allows the generation of graphic editors. It is based on EMF (Eclipse Modeling Framework) and GEF (Graphical Editing Framework). The created model with our realized editor can be serializable as XML, which is compliant with the task meta-model.

The realized tool is a plug-in for the Eclipse platform of development. Fig. 4 presents the task meta-model and the task model associated with the sub-process "Ask for a credit card" realized by the customer. From level 1, we detail the sub-process and the elementary tasks for sub-process "Ask for a credit card". This request consists in logging in, then choosing the customer type (private individual or company), finally determining a form.

Concept (Meta-) Model. Each data listed in the analysis of the requirements is modelled in a Concept Model. Each concept is connected to one or several tasks in which it contributes to their realizations. The domain concepts can be physical or abstract entities from the real world having a final representation at the level of the interface. These concepts intervene in the realization of the user task.

The state of the art proposes diverse formalisms for the representation of Concept Model: the entity-relation model [4] and the UML class diagram [14]. Our method suggests a concepts model based on UML class diagram. Fig. 5 proposes a simplified concept meta-model for an UML class diagram and a concept model illustrated by the case study. For example, the "Customer" represents a "Private Individual" or a "Company". The "Customers Responsible" manages the information connected to one or several customers. The "Responsible Service" manages the weight "WeightField" and the card types of "Card" in the banking institution.

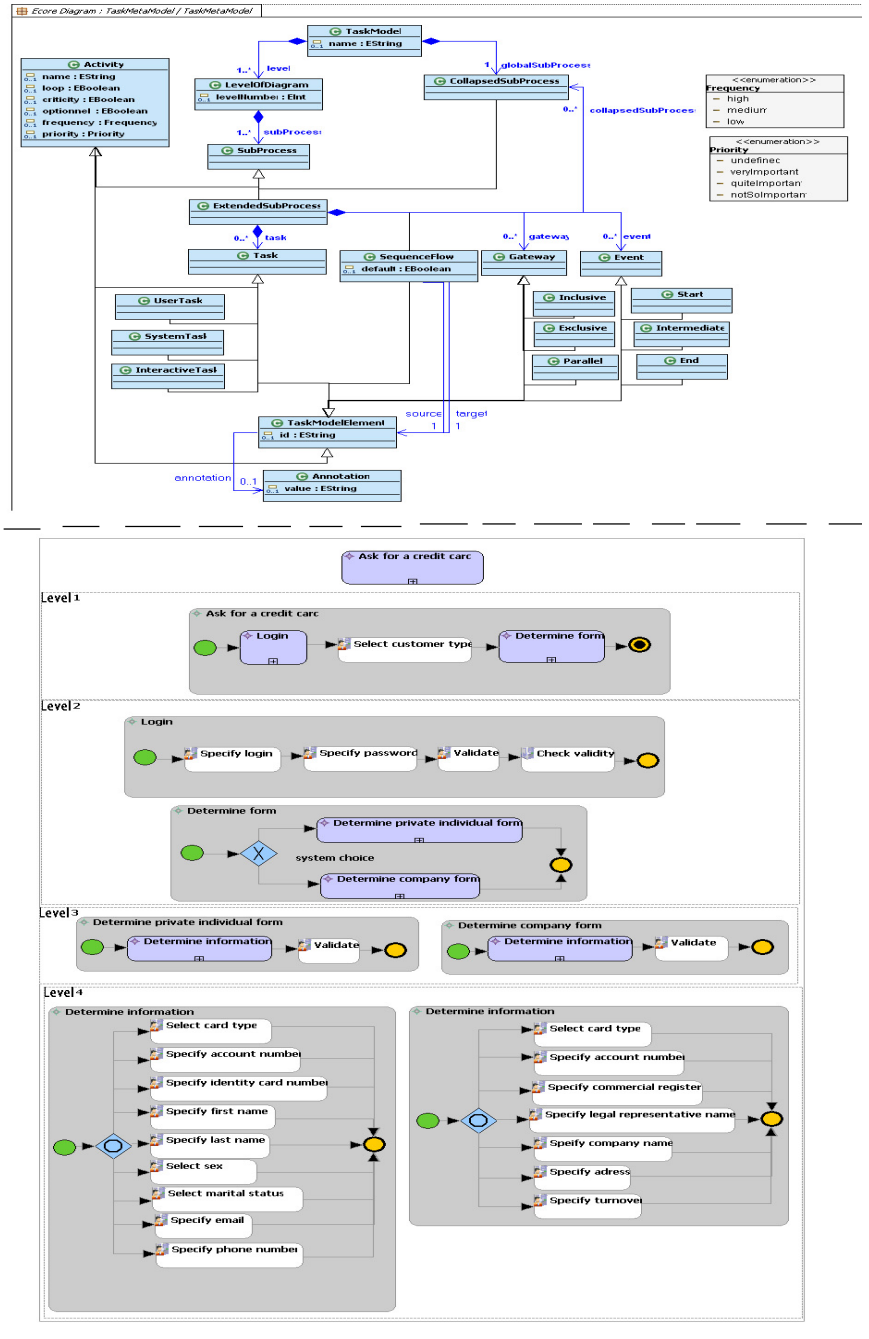

Fig. 4. Task Meta-model and Task Model for the possession process of the credit card

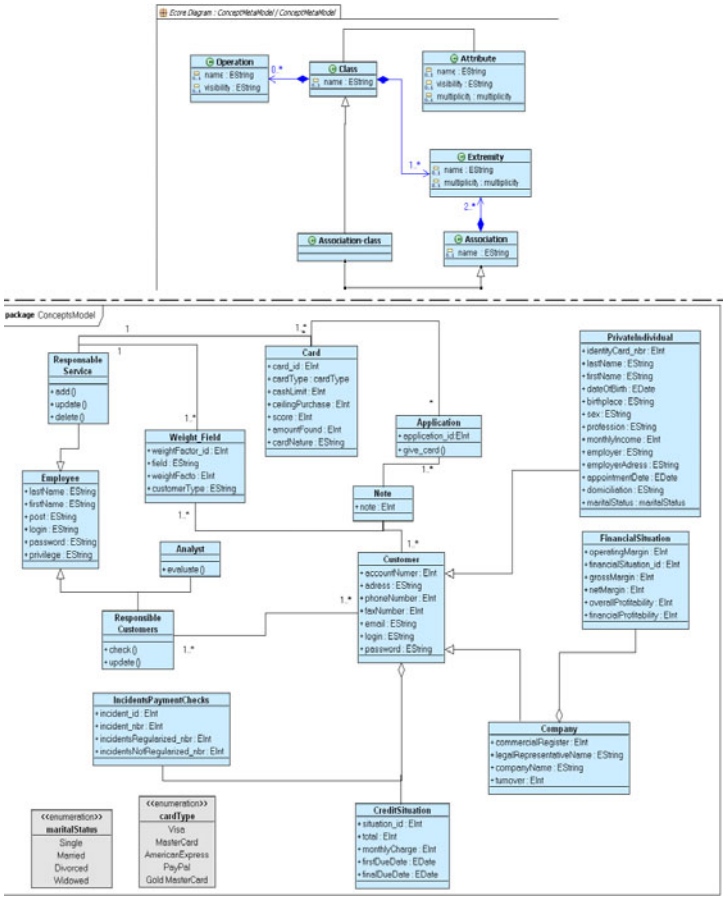

Fig. 5. Concept Meta-model and Concept Model for the possession process of the credit card

Abstract User Interface (Meta-) Model. In the literature, the abstract user interface is defined in several ways. In fact, Thevenin [32] defines it as a set of interconnected workspaces. A workspace is an abstract structure in which an interaction is organized. The connection between workspaces is made according to links between the tasks and the domain concepts.

In our approach, the Abstract User Interface (AUI) 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 appeal to a static model of the interactions [6]. Aiming at applying a model-to-model transformation, we have refined the static model of the interactions of [6] in the form of two meta-models: The AUI and CUI meta-models. AUI meta-model which is shown in [6] describes the hierarchy of the abstract components "UIComponent" corresponding to logical groups of interactions "UISpace". The

modelling of the abstract interface of an application is then made by one or several "UIGroup" which model containers forming coherent graphic elements (a window in a Windows environment, for example). Each "UIGroup" consists of one or several "UIUnitSuit" and/or "UIUnit". A "UIUnit" gathers a set of interaction elements which cannot be separated from a logical business point of view of the application (a treatment form for example), a "UIUnit" can include one or several "UISubUnit". The advantage of this modelling is to allow the creation of the application by assembling the existing elements, resulting in a strong reusability. The AUI is expressed by means of the BPMN notation, through the use of a similar formalism, which is in harmony with the established workflow and the task model. Fig. 6 shows also the abstract user interface for the possession process of the credit card. This interface contains a "UIGroup" associated with the global sub-process "Ask for a credit card". This "UIGroup" gives access to two "UIUnitSuit" ("Login" and "Determine private individual form") and "CollapsedUIUnit" ("Select customer type").

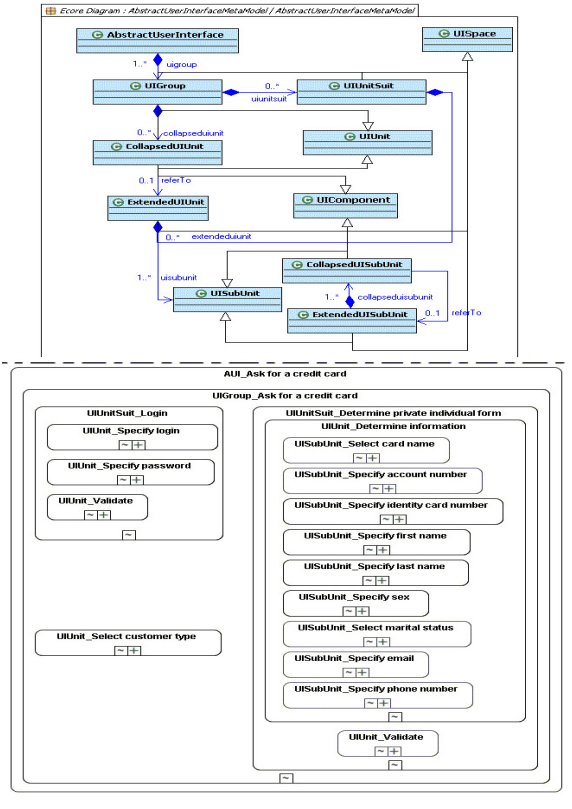

Fig. 6. AUI Meta-model and AUI for the possession process of the credit card (case of a private individual customer)

Concrete User Interface (Meta-) Model. The Concrete User Interface (CUI) is deduced from the Abstract User Interface (AUI) to describe the interface in terms of graphic containers, interactors and navigation objects. It is also expressed through the BPMN notation. The CUI meta-model extended from the static model of the interactions of [7] is presented in Fig. 7. It consists of one or several windows presented in the meta-model by the "UIWindow" class. The "UIPanel" class allows the modelling of the possible hierarchies of containers. The interactors presented by the "UIField" class of the concrete interface are classified according to their types in three groups: "UIFieldMultimedia", "UIFielData" and "UIFieldControl". The CUI presented in Fig. 7 shows a possible concretization of AUI (Fig. 6) for a small-screened platform.

Platform (Meta-) Model. Aiming at generating plastic interfaces, the platform meta-modelling has become a necessity in this work. Although most of the work on plastic UI made adaptation to the platform, the latter remains without a complete and detailed meta-model. The existing approaches only describe it at a high level of abstraction or describe only 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 inter-actor where the activation of the actions will be made vocally.

Fig. 8 presents our platform meta-model. Generally, the platform consists of:

- Calculation resources represented in Fig. 8 by the "ComputationalCapacities" class. These resources include not only the material aspect, such as the memory or processor but also software aspect as the supported operating system;
- Interaction resources which 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. 8) and the output devices (OutputDevice class in Fig. 8). Certain devices inherit both classes and are thus input/output devices, such as the touch screen. As concrete example, in Fig. 8 we give also the tree-based description of "iPAQ HX2490 Pocket PC" realized by EMF-based editor.

Final User Interface (Meta-) Model. The Final User Interface (FUI) is operational i.e., the user interface works on a specific platform deploying a programming language. In our approach, the acquisition of the FUI is made by a "Model to Code" transformation type. In fact, we correspond to each CUI component a final representation using a particular tool box. Some work is in progress to produce HTML and Swing UIs. Fig. 9 presents some sketches for the "iPAQ HX2490 Pocket PC" platform of the PDA family.

4.2 Transformations Rules for Plasticity

The proposed approach of UIs development is defined by a series of models transformations, each of which takes input models and produces output models, until

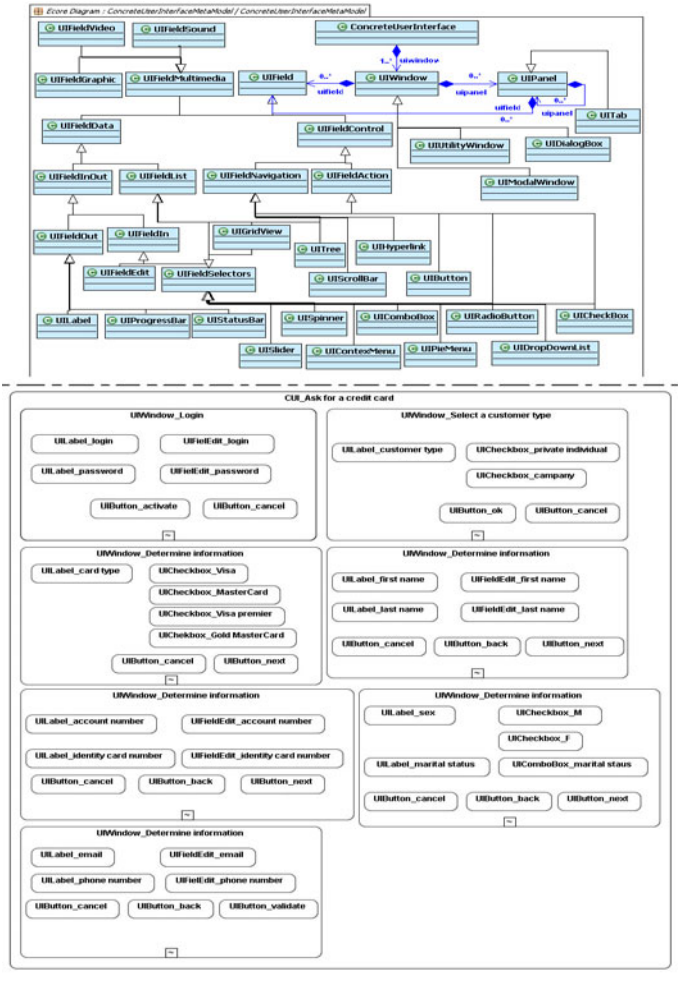

Fig. 7. CUI Meta-model and CUI for the possession process of the credit card (case of a private individual customer)

the obtaining of the final interface. This allows the HCI models (Concept Model, Task Model, AUI and CUI) to pass to the productive stage, which makes these models dependent on one another. The transformations are made explicit by the MDE; they are the core of the models engineering. Indeed, a transformation indicates a set of rules denoting the passage of a source model to a target model. For our approach, we used the Kermeta (Kernel meta-modelling) transformation language. Kermeta is created to allow the description of the meta-models as well as the definition of the constraints. It can also be considered as a language of action to give the operational semantics of the meta-models.

We propose two transformation modules. The first module is TMa2AUI having as source model the task model annotated by the concepts, allows the generation

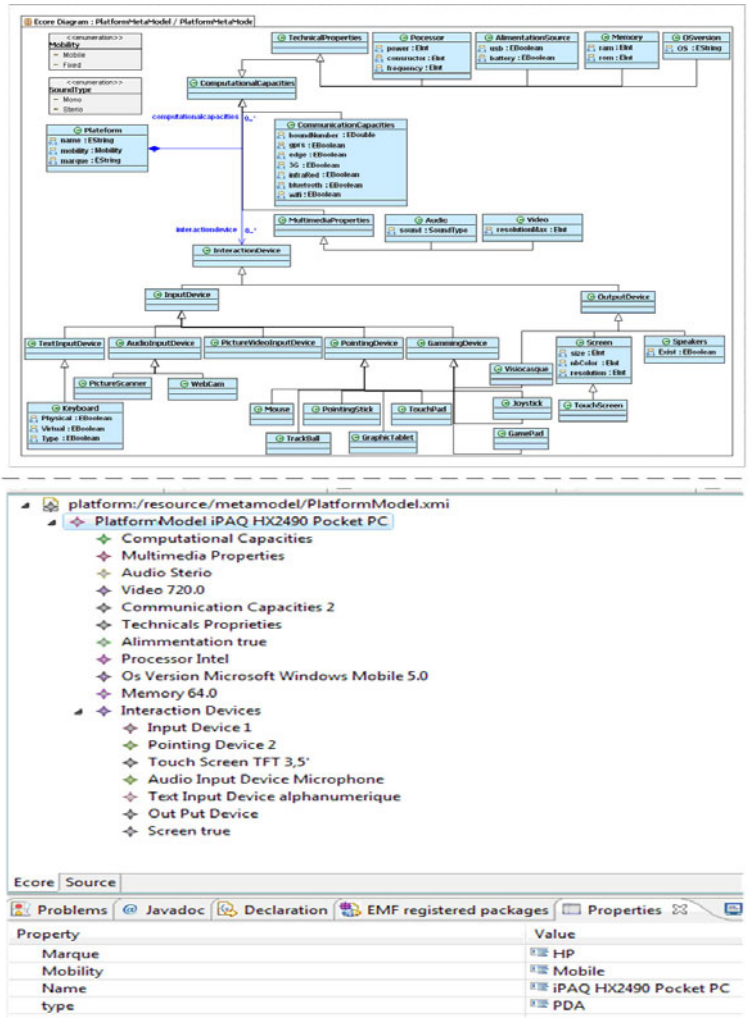

Fig. 8. Platform Meta-model and tree-based description of "iPAQ HX2490 Pocket PC"

of the AUI. Before introducing the task model into the transformation, it must be annotated by the domain concepts. This annotation allows the description of the matchings and the links between the concept model and the task model. In fact, the tasks treat the Concepts which are necessary for their implementation. In the approach of [27], the matchings between the models are done thanks to a mapping model which identifies several types of relations. In our approach, we exploited the artefact "Annotation" to attach to each elementary task a set of attributes and/or specified methods. For example, Fig. 10 shows the "Select card type" task annotated by the domain concept "cardType".

The TMa2AUI transformation consists in creating an abstract user interface, their containers (UISpace) and abstract components (UIComponent) from the

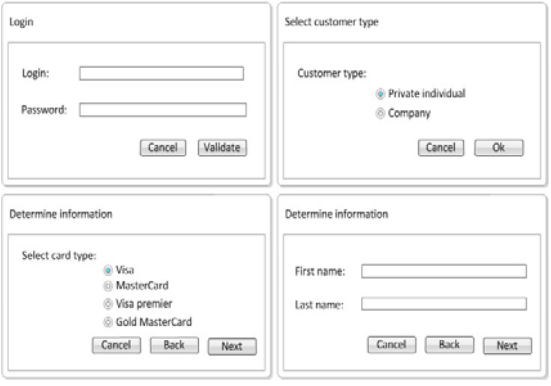

Fig. 9. Sketches for the possession process of the credit card (case of a private individual customer)

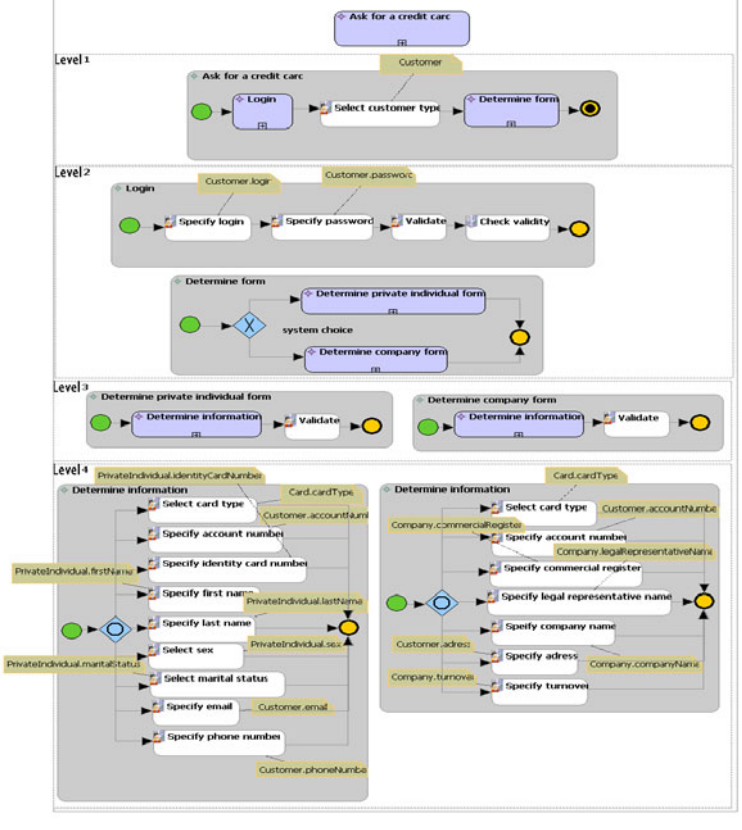

Fig. 10. Task Model "Ask for a credit card" annotated by the Concepts

annotated task model. In practice, this transformation can be implemented in Kermeta language by the following four stages:

- Creation of the application: creation of the application in the target model "AbstractUserInterface" by the "TaskModel" of the source model;
- Creation of the "UISpace";
- Creation of the "UIComponent";

To apply these stages, a set of rules is established. As an example, the extracted code below shows the instructions of "LevelsTreatment" method. It allows the creation of the "UIUnitSuit" container. Firstly, we have to recuperate the "CollapsedSubProcess" (CSP) from the "ExtendedSubProcess" (ESP) belonging to level one of the task model through the use of the "getCSPfromESP" method. Secondly, we have to create the "UIUnitSuit" instance in order to initialize and add them to the corresponding "UIGroup" (uig). Finally, we resort to the "ExtendContainerTreatment" method which is a reflexive method allowing the creation of the corresponding abstract containers of a container father.

```
operation LevelsTreatment(level : LevelOfDiagram, uig :UIGroup
,inputModel : TaskModel )
  is do
    getCSPfromESP(c).each{k|
    var uius : UIUnitSuit init UIUnitSuit.new
    uius.name := k.name
    uig.uiunitsuit.add(uius)
    ExtendContainerTreatment(level,uius,inputModel)
  end
```

The result of the transformation is an XMI file, which can be visualized by means of the AbstractUserInterfaceEditor developed with the Eclipse GMF plug-in which is based on the defined meta-model (Fig. 6). The first transformation allows having independent UIs of any modality of interaction and any implementation technology. However, the second module of transformation AUI2CUI which allows the generation of the CUI from the AUI is parameterized according to the target platform. This parameter setting presents the starting point of a standard description to a contextual description taking into account platform criteria. To do so, we build on the parameterized transformations defined by [34]. Vale [34] describes a parameterized transformation within the framework of the model driven engineering for a contextual development. The methodology proposed by [34] consists in defining the correspondences between the context model and the Platform Independent Model: PIM (Platform Independent Model) to define a CPIM (Contextual PIM). Then, an ordinary MDE transformation is used to define the CPSM (Contextual Platform Specific Model). The correspondences are assured by a parameter setting of the transformation. Its basic principle is to take into account the properties of the context during the transformation rules specification.

Based on the principle of contextualisation evoked by [34], we can use "the parameterized transformation" in the field of UI Engineering to consider the

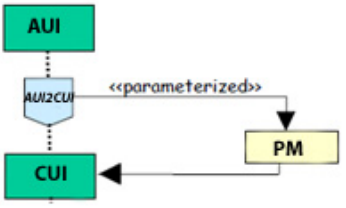

Fig. 11. AUI2CUI parameterized transformation

context of use. Such a transformation requires a triplet of models <Source, Target, Parameter>. The source model and the target model represent models of functional description (initial models) while the parameter model plays the role of a context model served for the contextualisation of the target model. Fig. 11 clarifies the transformation principle parameterized in our scenario. The parameter setting of the transformation leads to a strategy generation of UI adaptable to the context of use. We apply this parameter setting at the level of the AUI2CUI transformation. The context of use, taken into consideration, concerns particularly the platform.

The extract of the code below shows the operation "transform" which takes as a parameter two models: the input model (InputModel) which represents the AbstractUserInterface and the PlatformModel as adaptation model.

```
operation transform(inputModel :AbstractUserInterface
,adaptModel :PlatformModel) :ConcreteUserInteface is do
(The rest of code)
```

The generation stages of the Concrete User Interface lean strongly on the work of [32] and [20]. The AUI2CUI transformation can be implemented in Kermeta language 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 the transformation of an AUI into a CUI. 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 or Turn) and the platform properties (screen size, presence or absence of a keyboard, a mouse, a microphone etc...).

The UIField class of CUI meta-model presents a generalization of the various forms of interactors. The extract of the following code transforms every abstract component of the "CollapsedUIUnit" type into a "UIField" and appeals to the

"UIFieldTreatment" 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. We treat two cases as examples:

- If we have a keyboard or a touch screen, the "UIField" will be specialized in a "UIFieldIn" and a "UIFieldStatic".
- Else if we have input device of type "microphone" or "visiocasque", the "UIField" will be specialized in a "UIFieldSound".

```
operation TransformationTreatment(aui : AbstractUserInterface
,uiw : UIWindow ,p : PlatformModel)
  is do
    getAllCollapsedUnit(aui).each {cui|
    UIFieldTreatment(aui,p, cui,uiw)}
  end

// UIField specification
operation UIFieldTreatment( inputmodel : AbstractUserInterface,
paramModel : PlatformModel, cui : CollapsedUIUnit
, uiw : UIWindow)
  is do
  // recovery of annotation
  var lnk : Link
  lnk := getLinks(inputmodel)
  .detect{c|c.uicomponent.name== cui.name}
  var nat : Nature init cui.nature
  var tp : AnnotationType init lnk.uicomponentannotation.type

  if (MouseExist(paramModel) and ScreenExist(paramModel)
  and KeyboardExist(paramModel)) or (TouchPadExist (paramModel)
  and ScreenExist(paramModel) and KeyboardExist (paramModel))
  or TouchscreenExist(paramModel) then

  // Treatment of abstract component of type "Specify"
     if (nat == Nature.Specify) then
        createFieldIn(uiw,cui,lnk) // Creation of UIFieldIn
        createUILabel(uiw,cui,lnk) // Creation of UILabel
     end
rest of code
  else if VisiocasqueExist(paramModel)and MicroExist(paramModel)
  then
     createFieldSound(uiw,cui,lnk) // Creation of UIFieldSound
rest of code
  end
```

5 Conclusion and Perspectives

In this paper, we have presented a methodology for the development of the plastic UI of an Information System. Based on a BPMN workflow model, the processes are determined. Each process is detailed in a task model which follows a series of models transformations, according to an MDE approach to have the final UI. We proposed an extension in the BPMN notation to support the task modelling. To apply "model to model" transformations, we set up three meta-models: Task meta-model, Abstract User Interface meta-model and Concrete User Interface meta-model. The characteristic of the interface adaptation to its context of use was among our objectives. In order to reach them, we proposed a platform meta-model describing the material and software constituents of the interaction platform.

Our approach is distinguished from the existing approaches by:

- The use of a standard notation for the modelling of the majority of our approach models.
- According to the literature, the task modelling is, for the first time, made through the BPMN notation.
- The proposition of a complete and detailed meta-model for the platform. Encountered by new platforms, a definition of a model for this platform will be enough. So, our transformations rules are generic.

We foresee multiple perspectives for our work. These perspectives concern a meta-modelling of the environment and the user and the integration of the ergonomic properties in our transformations.

References

1. Allaguy, M., Trabelsi, W.: Crédit Scoring: L'octroi des cartes bancaires. In: Mémoire de fin d'étude, Institut des Hautes Etudes Commerciales de Carthage, Université de 7 Novembre á Carthage, Tunisie (2009)
2. Bézivin, J., Blay, M., Bouzeghoub, M., Estublier, J., Favre, J.-M.: Action spécifique CNRS sur l'Ingénierie Dirigée par les Modèles, Rapport de synthèse, janvier (2005)
3. Bézivin, J.: In Search of a Basic Principle for Model-Driven Engineering. Journal Novatica, Special Issus (2004)
4. Bodart, F., Hennebert, A., Lheureux, J., Provot, I., Sacré, B., Vanderdonckt, J.: Towards a systematic building of software architecture: The TRIDENT methodological guide. In: Proceedings of 1st Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS 1995, Vienna. Springer, Heidelberg (1995)
5. BPMI, Business Process Modeling Notation version 1.0., `http://www.bpmn.org`
6. Brossard, A., Abed, M., Kolski, C.: Modélisation conceptuelle des IHM: Une approche globale s'appuyant sur les processus metier. Ingénierie des Systèmes d'Information - Revue des sciences et technologies de l'information, Numéro 2007/5, France, pp. 69–108 (2007)
7. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15(3), 289–308 (2003)

8. Daassi, O.: Les Comets: une nouvelle génération d'Interacteurs pour la plasticité des interfaces Homme-Machine. Thèse de doctorat de l'université Joseph Fourier (2007)
9. Davenport, T.: Process Innovation: Reengineering work through information technology. Harvard Business School Press, Boston (1993)
10. Favre, J.-M., Estublier, J., Blay-Fornarino, M.: L'ingénierie dirigée par les modèles. Hermès-Lavoisier, Paris (2006)
11. Favre, J.-M.: Toward a Basic Theory to Model: Model Driven Engineering. In: Workshop on Software Model Engineering, Wisme 2004, Lisbonne, Portugal (2004)
12. Fingar, P., Bellini, J.: The Real-Time Enterprise. Meghan-Kiffer Press, New York (2004)
13. GMF, Graphical Modeling Framework, `http://www.eclipse.org/gmf`
14. Griffiths, T., Barclay, P.J., Paton, N.W., McKirdy, J., Kennedy, J.B., Gray, P.D., Cooper, R., Goble, C.A., Silva, P.P.: Teallach: A Model-based user interface development environment for object databases. Interacting with Computers 14(1), 31–68 (2001)
15. Guerrero, J., Lemaigre, C., Gonzalez, J.-M., Vonderdonckt, J.: Model-Driven Approach to Design User Interfaces for Workflow Information Systems. Journal of Universal Computer Science 14(19), 3160–3173 (2008)
16. Guerrero, J., Vanderdonckt, J., Gonzalez, J.M.: FlowiXML: a Step towards Designing Workflow Management Systems. Journal of Web Engineering 4(2), 163–182 (2008)
17. Kleppe, A., Warmer, J., Bast, J.: MDA Explained-The Model Driven Architecture: Practice and Promise. Addison-Wesley, Reading (2003)
18. Koch, N., Kraus, A., Cachero, C., Melié, S.: Integration of Business Processes in Web Application Models. Journal of Web Engineering (2004)
19. Kristiansen, R., Traetteberg, H.: Model-Based User Interface Design in the Context of Workflow Models. In: Winckler, M., Johnson, H., Palanque, P. (eds.) TAMODIA 2007. LNCS, vol. 4849, pp. 227–239. Springer, Heidelberg (2007)
20. 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)
21. MDA, Model Driven Architecture, `http://www.omg.org/mda`
22. Mirbel, I.: Conciliating User Interface and Business Domain Analysis and Design OOIS 2003. Rivieres V.2003, pp. 383–399 (2003)
23. 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)
24. OMG, Object Management Group, `http://www.omg.org`
25. Salimifard, K., Wright, M.: Petri net-based Modelling of Workflow Systems: An Overview. European Journal of Operational Research 134, 664–676 (2001)
26. Sottet, J.S., Calvary, G., Favre, J.-M.: Ingénierie de l'Interaction Homme-Machine Dirigée par les modèles. In: Proceeding of IDM 2005, Paris, France, pp. 67–82 (2005)
27. 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 MoDELS 2006 Workshop on Model Driven Development of Advanced User Interfaces (October 3, 2006)

28. Sousa, K., Mendonça, H., Vanderdonckt, J., Rogier, E., Vandermeulen, J.: User Interface Derivation from Business Processes: A Model-Driven Approach for Organizational Engineering. In: Proc. of 23rd ACM SAC 2008, pp. 553–560. ACM Press, New York (2008)
29. Sukaviriya, N., Kumaran, S., Nandi, P., Heath, T.: Integrate Model-driven UI with Business Transformations: Shifting Focus of Model-driven UI. In: Proc. of MDDAUI 2005. CEUR Workshop Series, vol. 159 (2005)
30. Sukaviriya, N., Sinha, V., Ramachandra, T., Mani, S., Stolze, M.: User-centered Design & Business Process Modeling: Cross Road in Rapid Prototyping Tools. In: Baranauskas, C., Palanque, P., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4662, pp. 165–178. Springer, Heidelberg (2007)
31. Terrase, M., Savonnet, M., Leclercq, E., Grison, T., Beker, G.: Points de vue croisées sur les notions de modèle et métamodèle. In: IDM 2005 Premières Journées sur l'Ingénierie Dirigée par les Modèles, Paris 30 juin, 1 juillet (2005)
32. Thevenin, D.: Adaptation en Interaction Homme-Machine: Le cas de la Plasticité. Thèse de doctorat, Université Joseph Fourier, Grenoble I, pp. 212 (2001)
33. Traetteberg, H.: Integrating Dialog Modeling and Domain Modeling - the Case of Diamodl and the Eclipse Modeling Framework. Journal of Universal Computer Science 14(19), 3265–3278 (2008)
34. Vale, S., Hammoudi, S.: Context-aware Model Driven Development by Parameterized Transformation. In: Proceedings of MDISIS (2008)
35. 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)
36. Workflow Management Coalition, `http://www.wfmc.org/`