A global process for using model-driven approaches in user interface design

Sybille Caffiau, Patrick Girard
LISI / ENSMA
Site du Futurosope – Téléport 2
86961 Futuroscop Chasseneuil Cédex – France
{caffiaus, girard}@ensma.fr

ABSTRACT
In user interface design, model-driven approaches usually use a generative solution, which has obvious limitations, especially for advanced user interfaces. Based on strong associations between task models and dialogue models, we propose a global process, which facilitates the design of interactive applications conform to their models, with the including of a rule-checking step. This process permits either to start from a task model or a user-defined prototype. In any case, it allows an iterative development, in line with user-centered design standards.

Author Keywords
Task Model, Dialogue Model, Metamodel, Design Method, Model-Driven Approach.

ACM Classification Keywords
H5.2 [Information Interfaces and Presentation]: User Interfaces, User-centered design; H.1.2 [Models and Principles]: User/Machine Systems.

INTRODUCTION

Model-driven approaches have been promoted for years. Despite their great interest, they remain hard to use in the context of user-centered design, especially when novel interaction techniques are expected. Thus, several research works [1-3] used a generative approach to build user interfaces – mainly skeletons to be completed – from task models. Following the analysis we made in [4], we can argue that this approach has several limitations:

- Generating requires the addition of information in order to reach an operative stage of interfaces. This information can be added to high-level models, which then loose their original goal; so doing they become hard to understand and to use, because of their multiple semantics (for example, adding presentation information to task models results in adding new semantics to this model). The other way is to insert this information during the generating process. This second approach is for example used in TERESA [5] by the way of heuristics, which are applied during the process. This however results in a lack of understanding of such transformations by users.
- All considered research issues are concerned with classical WIMP[1] applications. The hierarchical structure of task models is used to build the interface navigation scheme. We demonstrated in [4] that introducing non menu-based interactions implies a non-automatic transformation of the dialogue.
- Generating is not easy to include in iterative design cycles such as HCI-adapted cycles. When changes are required, it is necessary to modify the high-level models, and to generate again a new skeleton, to be improved again by hand-made add-ons. Some results have been obtained around the definition of "round-trip engineering" [6, 7], but were not applied to HCI. More, this approach prevents the designers from starting from the prototype, which method is often used in post-WIMP design.

Our aim is to introduce a new way to use models in user interface design. Leaning on meta-models of one task model and one dialogue model, we wrote equivalence rules between such models. Then, we defined a new development cycle that can be used in a user-centered iterative approach.

In this paper, the context of the used models is first described. Then, an example of the meta-models is given, and equivalence rules are presented. In the third part, the proposed way to use these models in a development cycle is outlined.

CONTEXT OF THE STUDY: THE MODELS

The starting point of our work is the analysis from [4]. Whilst the generation appears to be too limitative, links between task models and user interfaces seem obvious. So, we decided to explore the possibility to establish strong links between task model and other models, and to consider exploiting said links in software design methods. For some reasons, which are external to our subject here, we chose the K-MAD model [8] as our task model.

In our laboratory, we have been working for some time on dialogue models and formal approaches in HCI. We introduced a software architecture model, H[4], which was

[1] Windows, Icons, Menus and Pointers

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

first dedicated to computer-aided systems. Coupled with that architectural model, we proposed a dialogue model, the hierarchical interactors [9], and developed tools to apply it [10]. Because of its hierarchical structure, the Hierarchical Interactor (HI) model appeared as the most suitable for our purpose. A previous study demonstrated the capacity to exploit these two models (K-MAD and HI) in HCI design [11].

Then, we defined the meta-model of these two models, which is published in [12]. We chose to use the EXPRESS language, an alternative to the OMG approach for meta-modeling. EXPRESS is a standard data modeling language for product data. It is formalized in the ISO Standard for the Exchange of Product model STEP (ISO 10303), and standardized as ISO 10303-11[2]. It is supported by complete verification tools, and allows a full expression of constraints [13].

PRINCIPLES

In this section, we give a short description of the K-MAD and HI models, and provide some examples of the meta-models.

The K-MAD model

The K-MAD model is a hierarchical model where tasks are decomposed in sub-tasks, with temporal operators describing the dynamics of the model. The description can be enhanced by the definition of objects and expressions (preconditions, post-conditions, and actions) to control the model dynamics more precisely. The semantics of these different elements is defined in details.

Figure 1 illustrates a sample of the EXPRESS definition of the central element of the model, the task.

The Hierarchical Interactor (HI) model

The HI model consists in a state machine model where the dialogue of the application is split into independent automata. Transitions are activated by tokens that represent user inputs or automaton productions.

The hierarchical organization of the model allows the automata to produce and consume tokens. The main advantages of this system are two-fold:

- Automata are independent from each other. They can be removed or added independently, without any change to others.
- Tokens are the key elements of the model. As they can refer to both user entries and automaton productions, they break the binding between user inputs and transitions. This allows to consider the dialogue at the level of abstract level one wants. This is particularly important when post-WIMP interaction techniques are used.

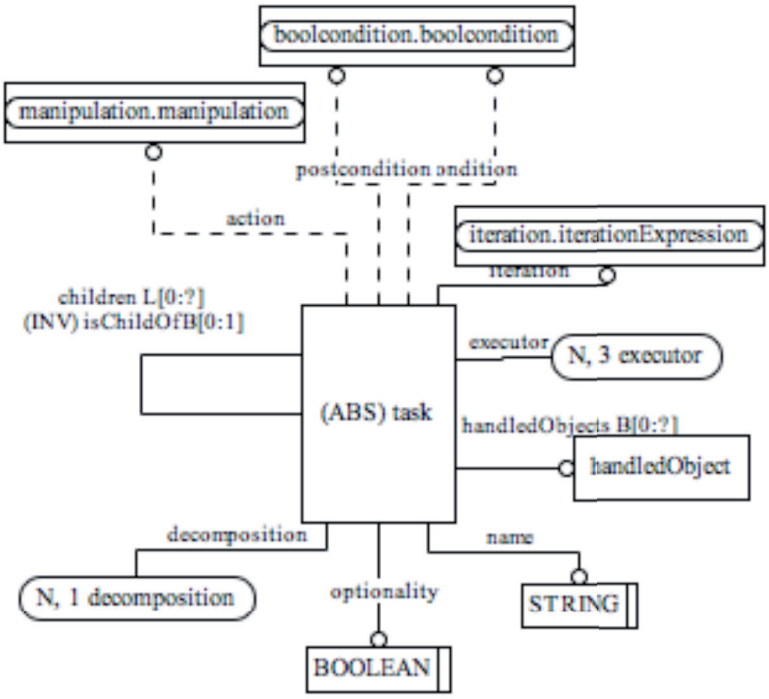

Figure 1: EXPRESS definition of the Task entity (partial)

Such as in advanced state machines, transitions may be guarded by expressions, which involve variables. They also can trigger actions. Figure 2 (next page) illustrates a transition meta-model.

Associations between models

The general philosophy of our approach is to take advantage of the hierarchical nature of the two models to establish strong associations between them.

The task/transition association

The first obvious association can be made between tasks from the task model and transitions from the dialogue model. This link has been largely used in the previous research works, but for us, the link is not a bijective link: because of the need for interaction facilities in applications, there can be more transitions than user tasks.

The compound-task/automaton association

The structure of the dialogue model encourages considering each task decomposition as equivalent to a specific automaton. The structure of the automaton must then be compatible with the dynamics described through the temporal operator of the compound task. Again, the dialogue may be richer than the simple translation of the temporal operator. Another consequence of this association is the equivalence between tokens and

[2] http://www.tc184-sc4.org/

compound tasks: each compound task may be achieved by the way of an automaton that produces a token that stands for the task achievement.

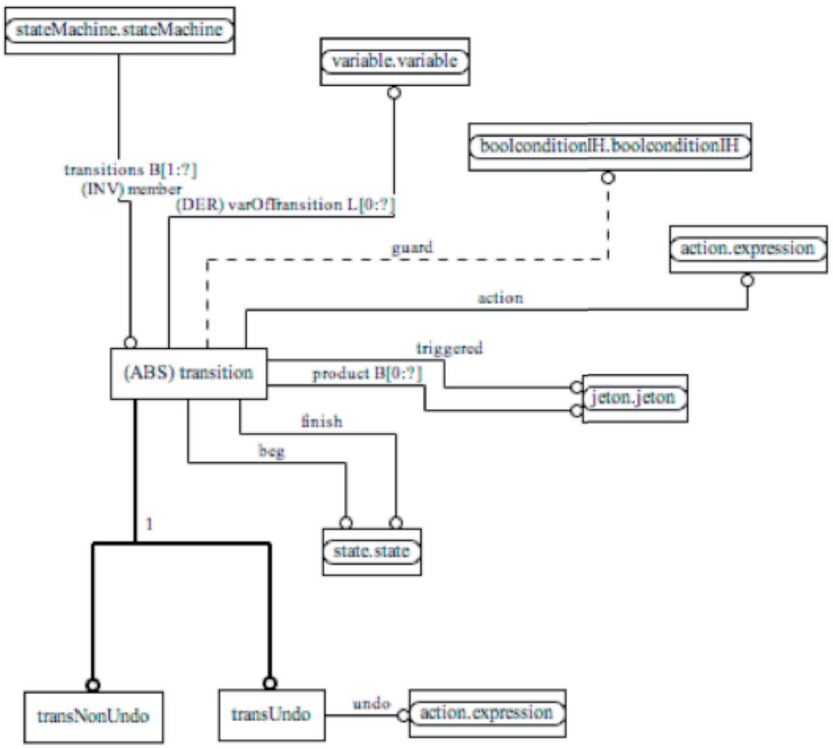

Figure 2: EXPRESS definition of the Transition entity

The object/variable and expression associations

Both task model and dialogue model use expressions, which manipulate objects/variables. This link is patent, but was not described in the previous works because the used task model did not formally consider objects and expressions.

Rules between models

Two kinds of rules can be established between the models [12], based on the previously defined associations.

The first kind of rules concerns the existence of logically associated entities in both models. For example, are there one token and one automaton for each compound task? Or is there one transition in the dialogue model for each task in the task model?

The second kind of rules relates to the semantics of the models. Are the semantics of the expressions we can find in each model equivalent? Is the navigation, which is allowed by the automata, consistent with the temporal decomposition of the tasks?

These rules can be exploited in two ways. They can be used in initial design to generate a skeleton of the dialogue, or they can be used in verification to state that two models are compatible. In that way, our work might be compared to [14]. We describe in the next section the different usages of this duality.

THE GLOBAL PROCESS

In this section, we describe the global process we propose to utilize the model-driven approach we describe above.

As previously claimed in the introduction, generating dialogue model from task models suffers from drawbacks; the most important of them is related to the iterative nature of user-centered approaches. When changes must be made in response to new or enhanced user needs, the generating process must be run again, and all hand-made changes in the interface are lost. We argue that, if a generation phase occurs, it must be restricted to a starting point; then, the process must be able to achieve without any further generation. The scenario schema we propose is as follows.

Assuming we are able to design, edit and verify each of both task and dialogue model. Each of these phases will be called "X-editing phase" thereafter. These phases may be realized independently from each other. Our model-driven approach consists in including these phases in a dynamic design process.

Optionally, one can start by a "task-editing phase", from which a starting skeleton for the dialogue model can be derived (e.g. generated, but only once). Either kinds of rules, existence rules and semantic rules, can be used to produce this skeleton. Then, the next phase consists in filling in the skeleton, in a "dialogue-editing phase". Adding specific dialogue elements, the dialogue model can be completed.

During this step, the two models can be confronted for detecting inconsistencies. By adding specific interaction elements to the skeleton, the designer might have changed the semantics of the model.

To reach this objective, the designer must associate the two models: some added dialogue entities might be related to task entities.

After analysis, depending on the result, different solutions can be applied:

- *Fail. The two models do not match. Some tasks are missing in the dialogue model.* The dialogue model

must be improved to take into account the whole task model.

- *Fail. The two models do not match.* The dynamics of the two models differ. The task model and/or the dialogue model must be modified.
- *Success. The two models match.* The system is now ready to being tested by users.

A user evaluation phase may result in new requirements, which may lead us to coming back to either dialogue or task modeling, and resuming the loop.

Figure 3 is a Petri net diagram that represents the global process. The process can start either from the Dialog-Editing Phase or the Task-Editing Phase. After rule checking, a failure results in redoing both Task and Dialogue Editing Phases. If problems are detected with usage or interaction during user evaluation, the process must also be repeated.

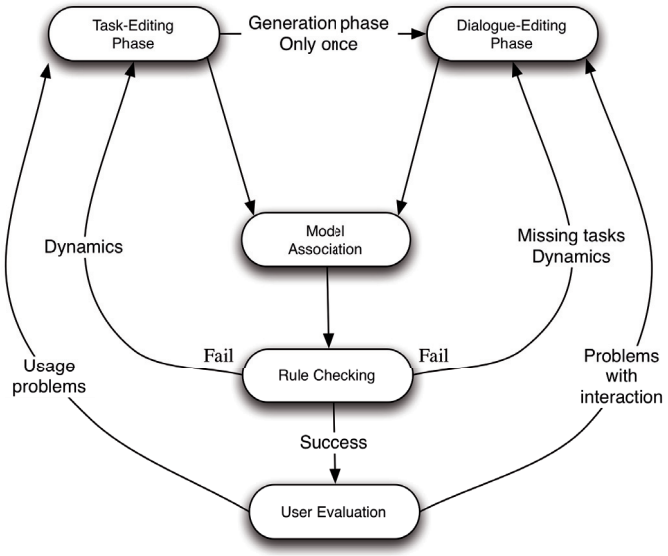

Figure 3: The global process.

CONCLUSION

In this paper, we present a global process to use a model-driven approach in user interface design. This process uses rules that allow to check the validity of task models and dialogue models. Moreover, this process is compliant to user-centered approaches that promote iterative design.

REFERENCES

1. Mori, G., F. Paternò, and C. Santoro, Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. IEEE Transactions on Software Engineering, 2004: p. 507-520.
2. Luyten, K., et al. Derivation of a Dialog Model from a Task Model by Activity Chain Extraction. in DSV-IS'2003. 2003. Funchal, Madeira Island, Portugal: Springer-Verlag. p. 203-217.
3. Wolff, A. and P. Forbrig. Deriving User Interfaces from Task Models. in MDDAUI'09. 2009. Sanibel Island, USA: CEUR-WS. p. 4.
4. Caffiau, S., et al. Generating Interactive Applications from Task Models: a Hard Challenge. in TAsk MOdels and DIAgrams (TAMODIA). 2007. Toulouse, France: Springer Berlin/Heidelberg. p. 267-272.
5. Berti, S., et al. TERESA: a transformation-based environment for designing and development multi-device interface. in Conference on Human Factors in Computing Systems - CHI'04. 2004. Vienna, Austria: ACM NY. p. 793-794.
6. Hettel, T., M. Lawley, and K. Raymond. Model Synchronisation: Definitions for Round-Trip Engineering. in Theory and Practice of Model Transformations, ICMT 2008. 2008. Zürich, Switzerland: Springer. p. 31-45.
7. Sendall, S. and J. Küster, Taming Model Round-Trip Engineering, in OOPSLA Workshop on Best Practices for Model Driven Software Development. 2004: Vancouver, Canada.
8. Lucquiaud, V. Proposition d'un noyau et d'une structure pour les modèles de tâches orientés utilisateurs. in 17th French-speacking conference on Human-computer interaction. 2005. Toulouse. p. 83-90.
9. Depaulis, F., et al., Le modèle d'architecture logicielle H4 : Principes, usages, outils et retours d'expérience dans les applications de conception technique. Revue d'Interaction Homme-Machine, 2006. 7(1): p. 93-129.
10. Depaulis, F., S. Maiano, and G. Texier. DTS-Edit : an Interactive Development Environment for Structured Dialog Applications. in CADUI'02. 2002. Valenciennes (France): Kluwer Academics. p. 75-82.
11. Caffiau, S., et al., Hierarchical Structure: A Step for Jointly Designing Interactive Software Dialog and Task Model, in Human-Computer Interaction. Novel Interaction Methods and Techniques, Springer, Editor. 2009, Springer: Berlin. p. 664-673.
12. Caffiau, S., Approche dirigée par les modèles pour la conception et la validation des applications interactives : une démarche basée sur la modélisation des tâches, in LIIS/ENSMA. 2009, Poitiers: Poitiers. p. 240.
13. Dehainsala, H., et al. Ingénierie dirigée par lesmodèles en EXPRESS : un exemple d'application. in IDM. 2005.
14. Kavaldjian, S., et al. Transformations between Specifications of Requirements and User Interfaces. in MDDAUI'09. 2009. Sanibel Island, USA: CEUR-WS. p. 4.