Generating Interactive Applications from Task Models: a Hard Challenge

Sybille Caffiau[1,2], Patrick Girard[1], Dominique Scapin[2], Laurent Guittet[1]

[1] LISI, Téléport 2-
1 avenue Clément Ader,
86961 Futuroscope Cedex, France
{caffiaus, girard, guittet}@ensma.fr

[2] INRIA, Domaine de Voluceau -
Rocquencourt- B.P.105,
78153 Le Chesnay, France
{Dominique.Scapin}@inria.fr

Abstract. Since early ergonomics, notations have been created focusing on the activities, jobs and task descriptions. However, the development of a wide variety of devices led to the generation of different interfaces from the same description of the tasks. The generation of complete current interfaces needs different types of information, some of which are not represented in usual task models. The goal of this paper is to present information that seems to be lacking in the task models.

Keywords: Task models, generation.

1 Introduction

Since early ergonomics, activities, jobs and task descriptions have been the center of any ergonomic diagnosis for assessment, evaluation and eventually for design and redesign. Lots of efforts were dedicated to data gathering, such as interviewing methods, and to identify issues with cognitive tasks (e.g. in air traffic control, nuclear power plants, etc.). Models for description, namely task models, have been then published and used [1, 2]. Tools supporting these models were later on developed, often not usually formal enough to allow full simulation and reuse of data.

Benefits from task-based modeling are nowadays largely reported in research [3]. Validation appears as the first aim of task-based approaches. In order to facilitate validation, some approaches, the model-based systems (MBS) [4], were developed to product user interfaces (UI) from models (of whom one is task models). More, the development of numerous devices and platforms requires to product UI capable of adapting to the context of use [5]. In order to design these kind of UIs, one strategy is to derive different UIs for several platforms from the same task model containing common information. This approach has been followed by ARTStudio [6], TERESA [7] and Dygimes framework [8]. After several steps, they produce final UIs adapted for a particular platform.

All these approaches are based on the generation of (all or a part of) UIs. However, during the different steps of generations, somme information are added (by users or

tools). Thus, the initial task model is modified. Then, validation according initial task models becomes complicated.

This paper is based on both a literature survey and case studies (on an email system, on a medical system). Exploring these case studies from the task model data highlights four challenges imposing to complete task models: interface presentation, definition of task to dialogue model transformations, connecting tasks with errors and undo patterns, and finally support rich forms of interaction (post-WIMP). In this paper, we expose these challenges respectively in section 2, 3, 4 and 5.

2 Tasks and Interface Presentation

Task models does not naturally contain elements of interface presentation. However, presentation has to provide mandatory elements to allow users to perform the tasks specified in task models. This consistency between task model and presentation may be exploited to verify that presentation allow to perform the feasible tasks (according to the scheduling).

Presentation contains interactive objects (widgets) that users need to carry out their tasks, applying some recommendations [9] (for example, to complete fields in a form, the user may use a customary order). A way to arrange widgets is to lay them according to the order of corresponding tasks as defined in the task tree (TERESA behaves that way). This approach uses semantics of order in the task model description. These semantics are useful for the presentation, completing the operators semantic. Thus, tasks linked by the “concurrent” operator should be performed concurrently, but the usual order is the order of description.

Furthermore, application presentation has to adapt to the platforms taking into account their space constraints. However, deducting the space constraints from the task model is impossible. Therefore studied approaches either respect the definition of the position of every element in the windows (as an example ARTStudio) or place the widgets according to screen size constraints (as an example Dygimes). Taking into account space constraints may impose addition of actions and thus modify task model. For example, consulting a menu on a computer necessitates less actions than on a phone.

At last, one of the functions of the presentation is to present the required information in order to execute a task for the user. However, existing approaches do not integrate this functionality. Cognitive tasks are usually added before interactive tasks, in order for the user to define an action strategy. Frequently, the execution of these cognitive tasks requires data to be displayed. For example, one needs to access a list content to pick a name from it. Such requirements could be fulfilled using objects, but, in our knowledge, no researcher’s work has yet followed this trend of researches.

3 Tasks and Dialogue Control

In most architectural models for HCI, the dialogue controller plays a central role, split in two responsibilities. First, it must associate user actions and functional core

procedures and functions. Second, it must control the dialogue, e.g. the exchange between the user and the application. That last point stands for ordering possible user actions depending on functional core and interface states, which is closely related to temporal control.

Temporal Control. Dialogue and task model seems to be very closed as shown by simulation tools of some task model editors (such as ConcurTaskTrees Environment [3] (CTTE[1]) or K-MADe[2] [10]). These task model editors supply designers to select from a set of enabled tasks which they want to evaluate (to create scenarios). More, an evaluation of the sets of feasible actions has to be performed on the dialogue control of interactive applications. The evaluation of enabled task sets [3] and feasible actions intuitively appears to share close similarities, and are even sometimes considered as identical [8, 7].

Nevertheless, is it true that these two sets can be considered as identical? Due to the difference between the points of view concerning the application that these two sets represent, some differences between the feasible actions and the enabled tasks sets exist. For example, task models present tasks entirely performed by user in the set of feasible tasks whereas they may not correspond to actions. Thus, whilst task model can express that a task can be performed only when a user task was carried out (using enabled operator in CTT or sequential operator in K-MAD), it is impossible to translate this relation in actions.

Furthermore, passing from a set of enabled tasks to another is performed through the execution of specific tasks (amongst a set of enabled tasks), such as when a user ought to enter a text before performing another task. In that example, only the user knows when the text is completed, the interface knows the end of the execution of the task when the user executes the following task. Thus, the second task can be performed as soon as and only when the first task begins. In order to generate interface from task models, detecting when the execution of these specific task ends has to be done. How can this be automatically done?

Link between Tasks and Functional Core. Interfaces need to be linked with the functional core in order to enable functions and procedures to be performed. Through this link, the various available actions are translated according to previously executed actions. Some information concerning the linked tasks is needed for the translation process, and several variables are manipulated.

Task model links are used to represent task decompositions as well as temporal organization between tasks. In task models, links can be expressed between sister tasks or between a mother task and its daughters. However, task executions may be linked through other relationships. For example, when the execution of a task can interrupt a set of others, they are linked together even if they are neither sisters nor mother and daughter. Representing the relations between tasks in a task model is sometimes challenging whereas it is necessary to identify and exploit all relations to design the dialogue. A first approach lies in the use of the deactivation operator. However, if the use of this operator allows to represent some conditions concerning the execution of the tasks, it does not answer the deletion issue. The temporal

[1] http://giove.cnuce.cnr.it/ctte.html

[2] http://www-rocq.inria.fr/merlin/kmade/ http://kmade.sourceforge.net/download.php

operators are not satisfactory for a precise control, but K-MAD [11] proposes the use of objects and conditions to improve the description. The semantics of these objects and conditions allows to represent some relation between tasks (the use of the same objects) but not all (deletion). More, task models present the user's viewpoint, thus the defined objects are the ones manipulated by user (corresponding to the state-of-the-world objects) and not the ones only required by the system (ex: boolean). How to deduct these non real-world objects from task models while they are not manipulated by the user?

4 The Human Factor

A great particularity of users is that… they often make errors, mistakes, and also, change their mind. Task models are not a good support to handle user errors, and express ways to correct them. Even if they can be used efficiently to explain errors, we will see in this section that they are not the best way to express error correction.

Whereas task models usually describe user activities without errors (the intention of the user), interface need to be designed for (the task execution). Including errors in the model leads to very hard representations. In order to take into account invalid data entries, tasks are split into two tasks, one to treat the entry of a valid data and another to the entry of invalid entries. However, the task models do not explain how to choose between the execution of these tasks although this information is necessary in order to perform error protection [9]. How should we complete task trees to, at the same time, allow user error anticipation and keep model readability?

The user may change his/her goal while using the application. Then, the interface should allow to undo a user action, as well as to go back to some previous state. However, task models do not take very well into account "undo" tasks. One can think of adding an "undo" task to each user action. But this design raises two questions:

(1) Can every task be undone? For example, when the user sends an email (clicking on a button "send"), the system realizes the associated action. This is an action you cannot undo, as it is impossible to have an email back once sent.

(2) Does a same "undo" task allow several user actions? For example, is undoing a writing task undoing all that was writing or only undoing the last word?

5 Tasks and Interaction

Task models are not designed to indicate what interaction is used to perform tasks. However, to make a complete interface, it is necessary to know what interaction is used. In order to proceed with generation from task models information are added. Some approaches such as *Dygimes Framework* [8], associate atomic tasks with the widget that allows its execution. Interaction with this type of approach is only composed of possible interactions with widgets. Furthermore, it assesses a bijective link between atomic tasks and widgets, which is very reducing for interaction (tasks may be executed differently according to the chosen interaction instruments).

Completing or modifying task models becomes necessary due to the fact that an atomic task may be performed by a succession of interactive actions (using, for example, *drag and drop)*, and that knowing where the action is done (on a file, in the message field…) may be important. Adding this information in the task models signifies inserting data that do not belong to the abstraction level of task models.

Today, more and more applications use new interaction techniques, generally grouped under the name "post-WIMP", in order to enhance direct manipulation principles [12]. Following these principles leads to the fact that user actions and system responses are very close and to the deletion of intermediaries such as dialogue boxes [13]. The order of the task execution may be modified according to the chosen interaction or instruments. Thus, some tasks may be deleted or added.

Even if all interactions for a given task could be represented in task models, how can we indicate when a task may be performed by two different interaction techniques? In *SUIDT* [14], a concrete task is created for each interaction. For example, if a task can be performed either clicking a button or shortcutting ("Ctrl+S"), then the task is refined in two concrete tasks linked by the alternative operator. This design increases the number of concrete tasks. Furthermore, as previously stated, the interaction chosen may modify or delete completely a task. Then, modifications of the task model may be made at a higher level and is not limited to adding concrete tasks.

Moreover, the scheduling of tasks is very close to the dialogue of the application. However, adding the interaction may modify the dialogue itself. For example, moving or deleting a file are completely different tasks from the point of view of the task model. Nevertheless, with *Drag and Drop*, their beginning phases are merged. The user starts by clicking on the file, and drags it. At this stage, it is not possible to know what is his/her goal, moving or deleting. The goal appears when the document is dropped: at another place in the document for moving, out of the window for deleting. The equivalence between the task model and the dialogue model is broken.

6 Conclusion and Perspectives

Our study aimed to find challenges to generate interactive applications from task models. We have identified four different ones partially filled by generation approaches. Each of them reveal different ways for future work.

How to rely presentation to task models will be our first challenge. Particularly, we will study the use of the cognitive aspects of activity in order to check the correspondence between the user needs and the presentation.

Concerning the management of user errors, we established that task models can participate to prevent errors, but are not well adapted to correct them. Connecting task models to errors and "undo" patterns is, in our opinion, a promising research trend that will encompass our second challenge.

The dialogue control seems to be very close to task models, as shown in the simulations of task model editors and in [15]. Nonetheless, our study shown that it is not so easy. Complex task models cannot be completely derived towards dialogue models. Transformations must be defined. Being able to keep links during this

transformation is our third challenge. Even if generating full dialogue control from task models is impossible, this link requires to be exploited. We aim at using task models to help the design of the dialogue and to validate it. Then, we will study how to establish the communication between these two models using, for example the MDE (Model-Driven Engineering) approaches[3] as Metamodels.

At last, we conclude with the study of rich models of interaction that evolve into post-WIMP interfaces, opening new ways for transformations between models.

References

1. Balbo, S., Ozkan, N., Paris, C.: Choosing the right task-modeling notation: A taxonomy. In: The handbook of task analysis for human-computer interaction. D. Diaper and N. A. Stanton (Editors), Lawrence Erlbaum Associates, pp. 445-466 (2004)
2. Limbourg, Q., Vanderdonckt, J.: Comparing task models for user interface design. In: The handbook of task analysis for humain-computer interaction. D. Diaper and N.A. Stanton (Editors), Lawrence Erlbaum Associates, pp. 135-154 (2004)
3. Paternò, F.: Model-based design and evaluation of interactive applications. Springer (2001)
4. Pinheiro da Silva, P.: User interface declarative models and development environments : A survey. 7th Eurographics workshop on Design, Specification and Verification of Interactive Systems DSVIS 2000. Springer. pp. 207--226 (2000)
5. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., Florins, M., Vanderdonckt, J.: Platicity of user interfaces: A revised reference framework. Tamodia. Bucarest. pp. 127--134 (2002)
6. Thévenin, D.: Adaptation en interaction homme-machine : Le cas de la plasticité. Thesis. Université Joseph Fourier. p. 213 (2001)
7. Mori, G., Paternò, F., Santoro, S.: Tool support for designing nomadic applications. Intelligent User Interfaces (IUI'2003). pp. 141--148 (2003)
8. Luyten, K.: Dynamic user interface generation for mobile and embedded systems with model-based user interface development. Thesis. School of Information Technology, University Limburg, Diepenbeek, Belgium. p. 194 (2004)
9. Bastien, C., Scapin, D.: Ergonomic criteria for the evaluation of human-computer interfaces. Technical report, INRIA (1993)
10. Baron, M, Lucquiaud, V., Autard, D., Scapin, D.: K-made : Un environnement pour le noyau du modèle de description de l'activité. IHM'06, ACM Publishers. pp. 287-288 (2006)
11. Lucquiaud, V.: Proposition d'un noyau et d'une structure pour les modèles de tâches orientés utilisateurs. 17th French-speaking conference on Human-computer interaction, pp. 83--90 (2005)
12. Shneiderman, B.: Direct manipulation: A step beyond programming languages. In: IEEE Computer 16, no. 8, pp. 57--69 (1983)
13. Beaudouin-Lafon, M.: Instrumental interaction: An interaction model for designing post-wimp user interfaces. CHI, The Hague, Netherlands. pp. 446-453 (2000)
14. Baron, M., Girard, P.: Suidt : A task model based gui-builder. TAMODIA : Task MOdels and DIAgrams for user interface design. Inforec Printing House. pp. 64-71 (2002)
15. Navarre, D., Palanque, P., Paterno, F., Santoro, C., Bastide, R.: A tool suite for integrating task and system models through scenarios. Interactive systems design, specification, and verification (dsv-is'01). C. Johnson (Editor), Springer-Verlag, Glasgow, Scotland, UK. pp. 88-113 (2001)

[3] http://planetmde.org