Tool support for handling mapping rules from domain to task models

Costin Pribeanu
National Institute for Research and Development in Informatics
Bd. Mareşal Averescu Nr. 8-10,
011455 Bucharest, Romania
pribeanu@ici.ro

ABSTRACT

The success of model-based approaches to user interface design depends on the ability to solve the mapping problem as well as on the availability of tools able to reduce the effort of establishing and maintaining of links between models throughout the development life cycle. In this paper a tool supporting a small set of mapping rules is presented. The tool enables the designer to produce task model fragments at operational level based on the patterns of mapping between task and domain models. The task model fragments are generated in XML format and can be further loaded in task modeling tools like CTTE or Teresa.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design tools and techniques. H5.2 [**Information Interfaces and presentation**] User interfaces.

General Terms

Design, Human Factors, Languages.

Keywords

Model-based design, Task models, Mapping problem.

1. INTRODUCTION

The explosion of mobile and embedded systems is challenging the development of interactive systems able to run in different contexts of use. The model-based approach could be seen as a progressive derivation of user interface components from representations expressing relations between users, tasks, domain, environment, and technology. The strength of this approach relies on the separation of various models which are capturing the context variations. In turn, the generative power of these abstractions relies mainly on the mappings between models.

The mapping problem has been defined in [8] as a key problem for the gradual transformation of models from abstract to concrete level as well as for the mapping between different models on the same level of abstraction. Previous work in this area highlights the concern for preserving consistency between models along the progression from one model to another [2], elaboration of graceful degradation rules for multi-target user interfaces [3] as well as development of a description language and tools supporting the specification transitions [4].

This paper is presenting a small set of mapping rules between task and domain models and a tool supporting the automate derivation of task model fragments from the domain model. The tool enables the designer to integrate domain modeling results (object, attributes and relationships) into task models that are developed by using the CTT notation [5].

The rest of this paper is organized as follows. In section 2, we will briefly describe our task modelling framework and some general mapping rules between domain, task and presentation models. Then we will describe a tool supporting a set of rules that are covering a significant effort in a task-based design of user interfaces. The paper ends with conclusion in section 4.

2. THE TASK MODELING FRAMEWORK

Our model-based design framework is focusing on the relations between three models: task, domain and presentation. The purpose of modeling is to derive as much as possible from the user interface based on the mappings between the components of these models.

The basic element in the presentation is the abstract interaction object (AIO). We distinguish between information control AIOs (such as text boxes, check boxes or lists) and function control AIOs (such as buttons or menus). The user interface is structured into dialog units featuring various AIO configurations. The user is manipulating AIOs to change something in the domain model: objects, attributes and relationships between objects.

We identified three layers which are relevant in the task modeling for user interface design:

- A functional layer that results from mapping application functions onto user tasks, corresponding to business goals (such as clients or order management).
- A planning layer that results from the decomposition of functional tasks up to the level of unit tasks [1, 7], having a clear relevance for the user (such as adding a new client or updating the client address).
- An operational layer that results from the decomposition of unit tasks up to the level of basic tasks. A basic task has been defined in [7] as the lowest level task that is using a single interaction object, or a single external object or serves a communicational goal.

Figure 1 is illustrating various kinds of patterns of mapping that occur in and between task, domain and presentation models. The framework shows *horizontal* mappings between elements from different models as well as *vertical* mappings within the hierarchical structure of each model.

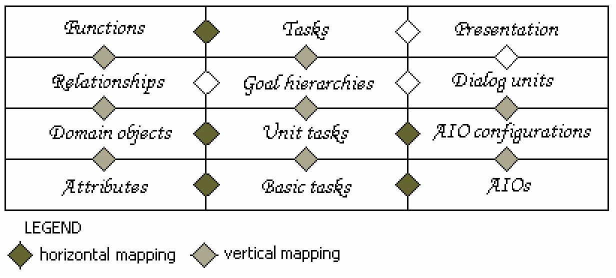


Fig. 1. Domain-task-presentation mappings

In this paper we will focus on mapping rules that apply to the lower levels of task and domain models, i.e. the mapping of domain objects and attributes onto unit tasks and basic tasks.

In order to illustrate our approach we will take an example: an application for data management in a trade company. The target task is the recording of new orders. In Figure 2, a task model representation using the CTT notation is given.

Tasks on the first decomposition level are corresponding to the business goals of the application: management of clients, products and orders. Each of them is further decomposed in tasks that correspond to the high level functions of the applications that support these goals. For the sake of simplicity and legibility of the representation, only the decomposition of the target task ("New order") is shown in Figure 2.

The task new order is a leaf in the functional layer and is further decomposed in the planning layer up to the level of unit tasks.

Unit tasks are further decomposed up to the level of basic tasks. Again, for the sake of legibility, only the task "New product" has been decomposed up to the level of basic tasks.

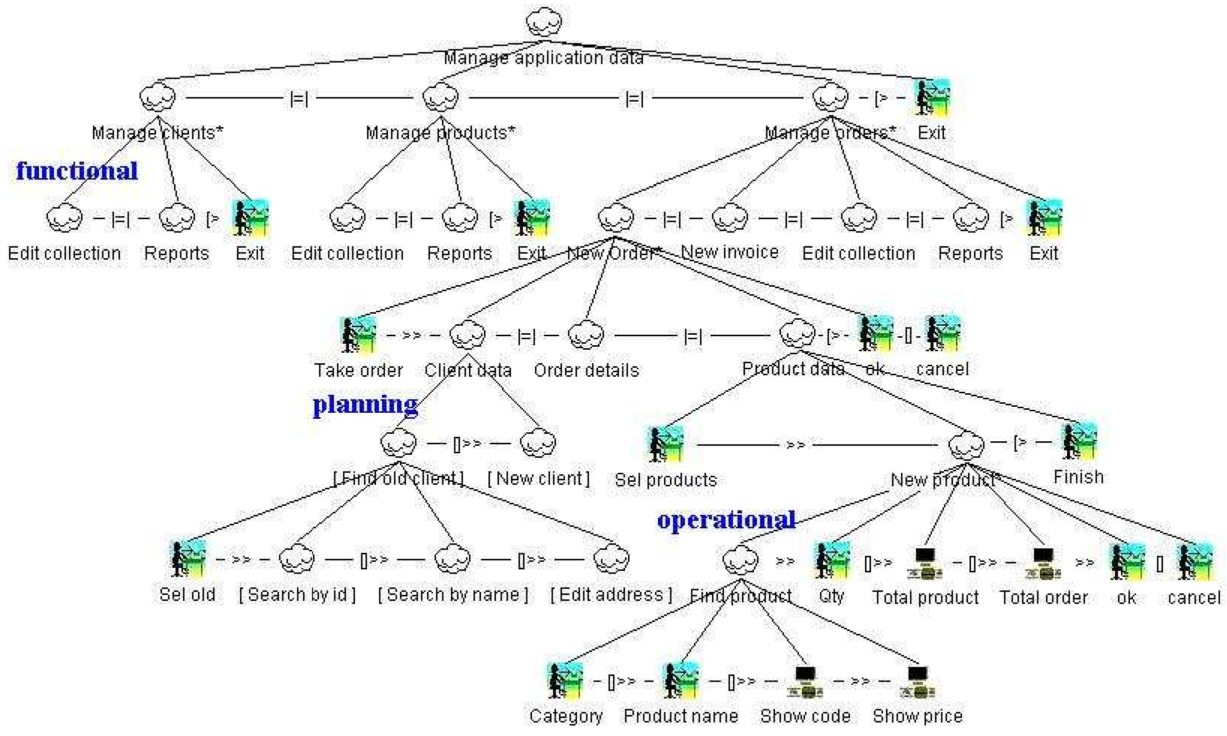


Fig. 2. An example illustrating the layered task modeling approach

In [7] it was shown that the operational task model suggests the first level of aggregation of abstract interaction objects into AIO groups. Interaction object groups, which have one or more information control AIO (for example, a text box or a list box) and one function control AIO (sometimes two, but the user could choose only one of them at a given time – for example buttons OK vs. Cancel) provide with a first level of structuring the interface. As such, they can be used as basic building blocks for the presentation model in a task-based design.

The goal of an information control basic task is the manipulation of a domain object attribute (such as display or edit). The mapping rule described below is well known in the model based design of user interfaces and has been widely used in early model-based approaches to user interface design.

MR1. Information control basic tasks in the task model are mapped onto domain object attributes in the domain model and abstract interaction objects in the presentation model. Attribute names are mapped onto AIO labels

The goal of a function control basic task is to trigger a transaction changing some attribute values in the domain model or to present them in the interface. Each basic task in this category is using a function control AIO. (Function control is sometimes termed as action control and the focus is on low level functions or commands provided by the user interface).

MR2. Function control basic tasks in the task model are mapped onto available commands on the target platform and abstract interaction objects (AIO) in the presentation model

Mapping rules MR1 and MR2 are the lowest level of horizontal mappings illustrated in Figure 1 on the last row of the table.

The operations performed on domain objects (such as display, new, update or delete) are mapped onto unit tasks. In Figure 2, the first basic task has an enabling role for the unit task. Usually, the task name is a concatenation of the enabling basic task name denoting the operation and the domain object name.

This task structure is a typical task pattern for data entry tasks carried on in a separate dialog unit and suggests a composition rule for this category of unit tasks. The mapping rule described below makes it possible the derivation of a great part of the task model (operational layer) from the application domain model. This is very useful when using task-based design tools for the computer-aided design of user interfaces.

MR3. Unit tasks corresponding to operations performed onto domain objects are usually starting with one (or two) function control basic task selecting the operation (and the object) and are ending with one or two function control basic tasks for the confirmation (or canceling) of task completion

Since this mapping takes the form of a composition, it could be further expanded in more detailed rules, following each type of operation. This way it is possible to automate the derivation of a great part of the task model from the domain model. For example, in the case of the task "New order" in Figure 2, the task model statistics provided by the CTTE tool shows a total of over 40 tasks for which the designer should manually specify the task model, including temporal relations (operators), attributes and objects for each task.

In the table in Figure 1, MR3 covers a vertical mapping in the task model (unit task-basic tasks) and a horizontal mapping between domain and task models (domain object-unit task).

3. TOOL SUPPORT FOR DETAILED MAPPING RULES

In order to illustrate more detailed mapping rules, we will use a simplified task notation that could be mapped onto the CTT notation, like in Figure 3. There are three types of basic tasks: two for information control (interactive and display only) and one for function control.

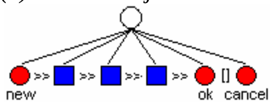
○ unit task	🟦 info control	🔴 function control	◻ display
abstraction	interaction	interaction	application

Fig. 3. A simplified task notation and the correspondence with the CTTE graphical notation

We identified five detailed mapping rules by applying MR3 to five operations performed onto domain objects. Each mapping rule is expressed bellow as a task pattern having a prefixed part, a sequence of information control basic tasks and a post fixed part. In three cases (b, c and d), a task for selecting the target object is needed before selecting the command.

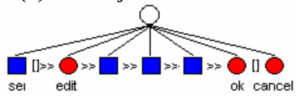
The process of computer-aided generation of full decomposition for unit tasks is illustrated in Figure 4. The designer selects the object and the object attributes that are relevant for the context of use. Then (s) he checks on the operations to be performed onto it. In the case of a search operation, (s) he will also select the search key attribute. The generated unit tasks are shown in the lower left list box.

(a) add new object



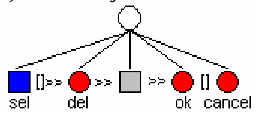
The user selects the "new" command and the object attributes are displayed with their default values and available for data entry. The user can confirm or cancel the transaction.

(b) edit object attributes



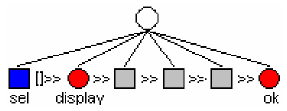
The user selects the object to be modified and then selects the "edit" command. The object attributes are displayed and available for data entry. The user can confirm or cancel the transaction.

(c) delete object



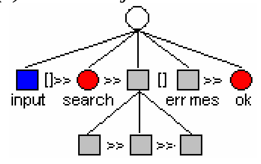
The user selects the object to be deleted and then selects the "delete" command. A shield message is displayed so the user could check once again if (s) he really wants to perform. The user can confirm or cancel the transaction.

(d) display object attributes



The user selects the object to be displayed and then selects the "display" command. The object attributes are displayed until the user confirm the visualization

(e) search object

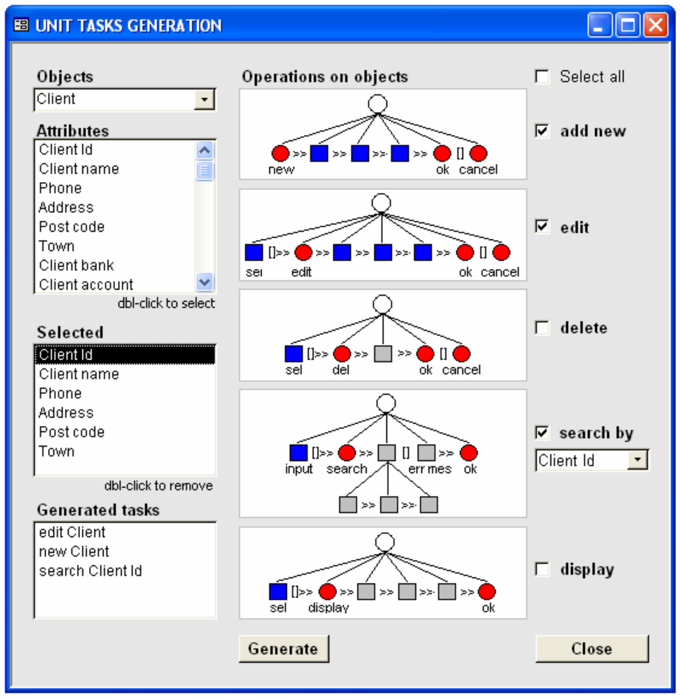


The user inputs the search key (attribute) and then selects the "search" command. If the search succeeds, then object attributes are displayed. Otherwise, an error message is displayed.

The designer can choose all the operations or only those that are relevant for the context of use. Moreover, (s) he can select only attributes that are relevant for the target context of use.

For example, according to the functional layer in Figure 2, in the context client data management, all operations on domain objects and all object attributes are needed while in the of recording a new order, only the operations that are checked in Figure 4 are selected. On another hand, the "search object" pattern is applied twice in this case (search by id and search by name).

In some situations, a manual post processing might be needed for the task model fragment generated by the tool. For example, the enabling basic task might not be needed, if the unit task is implicitly enabled. An example is the case of iterative tasks that are implicitly started and explicitly stopped, like in the case of "New product" in Figure 2. The user ends the iteration by selecting the "Finish" command.


UNIT TASKS GENERATION

Objects: Client

Attributes: Client Id, Client name, Phone, Address, Post code, Town, Client bank, Client account

Selected: Client Id, Client name, Phone, Address, Post code, Town

Generated tasks: edit Client, new Client, search Client Id

Operations on objects: ☐ Select all; ☑ add new; ☑ edit; ☐ delete; ☑ search by: Client Id; ☐ display

Fig. 4. Tool supporting mappings from domain to task model

The unit tasks are generated in XML format and are loaded in the CTTE tool [5] with the "Load CTT as XML" function. The generation process is producing a full specification (task attributes and interaction objects) according to the specification of domain object attributes in the domain model (a time consuming work if manually introduced with the CTT editor).

4. CONCLUSION AND FUTURE WORK

In our task-based approach, the task model is gradually developed from functional to planning and operational levels. In this paper we presented a tool supporting a small set of patterns of mapping between task and domain models at operational level. Further work is needed to extend these detailed mapping rules and to explore the mappings between goal hierarchies in the task model and relationships between domain objects in the domain model.

The mapping rules are preserving the consistency between domain, task and presentation models and make it possible the computer aided design of user interface. In this respect, the specification of domain objects is automatically transformed into a XML specification of unit tasks following the composition rules. Then the generated tasks are loaded in Teresa [6] or other tool supporting the computer-aided generation of the presentation. Since task variations play an important role when migrating from a target context of use to another, the computer aided generation of (an important part of) contextualized task models is a key facility for designers.

5. Acknowledgement

This tool has been developed during a scientific visit at ISTI-CNR Pisa. We gratefully acknowledge the support of SIMILAR NoE under FP6-507609 project.

References

[1] Card, S. K., Moran, T. P. and Newell, A.: The psychology of human-computer interaction. Lawrence Erlbaum Associates. (1983).

[2] Clerckx, T., Luyten, K. & Coninx, C.: The mapping problem back and forth: Customizing dynamic models while preserving consistency. Proc. of Tamodia 2004 (2004) 99-104.

[3] Florins, M. & Vanderdonckt, J.: Graceful degradation of user interfaces as a design method for multiplatform systems. Proceedings of IUI'2004. ACM Press (2004) 140-147

[4] Limbourg, Q. & Vanderdonckt, J.: Addressing the mapping problem in user interface design with USIXML. Proc. of Tamodia 2004 (2004) 155-164.

[5] Paternò, F., Mancini, C., Meniconi, S.: ConcurTaskTree: a Diagrammatic Notation for Specifying Task Models. In: Proceedings of IFIP TC 13 Int. Conf. on Human-Computer Interaction (Syndey, June 1997). Chapman & Hall, London (1997), 362–369

[6] Paternò, F. , Santoro, C. :One Model, Many Interfaces. Proceedings of CADUI'2002, Kluwer. 143-154.

[7] Pribeanu, C. & J. Vanderdonckt (2002) Exploring Design Heuristics for User Interface Derivation from Task and Domain Models. Proceedings of CADUI'2002, Kluwer,103-110.

[8] Puerta, A.R. & Einsesnstein: J. Towards a general computational framework for model-based interface development systems. Proceedings of IUI'99 (5-8 January 1999). ACM Press. (1999). 171-178.