An Approach to Task Modeling for User Interface Design

Costin Pribeanu

Abstract—The model-based approach to user interface design relies on developing separate models capturing various aspects about users, tasks, application domain, presentation and dialog structures. This paper presents a task modeling approach for user interface design and aims at exploring mappings between task, domain and presentation models. The basic idea of our approach is to identify typical configurations in task and domain models and to investigate how they relate each other. A special emphasis is put on application-specific functions and mappings between domain objects and operational task structures. In this respect, we will address two layers in task decomposition: a functional (planning) layer and an operational layer.

***Keywords*—**task modeling, user interface design, unit tasks, basic tasks, operational task model.

I. INTRODUCTION

Model-based approaches to user interface design are creating a promising framework for the development of interactive systems able to run in different contexts of use. Rather than starting from scratch, this approach makes it possible to develop separate models which are capturing context variations and to manipulate them in order to migrate from one context to another.

Models are used to capture design knowledge needed for the construction of the future user interface (UI). Main concepts abstracted into these models refer to users, tasks, application domain, presentation and dialog. User, task and domain models may be termed as contributing models since they are influencing the UI design process. In this respect, they are used for the derivation of presentation and dialog models. Model-based approaches which are giving the task model a leading role among the other models are also referred to as task-based approaches.

A key ingredient to the success of model-based approaches is the mapping problem related to the handling of relationships between models throughout the development life cycle. These relationships or mappings are of particular importance when we want to run a target application in different contexts of use while preserving usability.

This paper presents a layered approach to task modeling in the framework of model-based design of user interfaces. We will address the mapping problem as a rationale for the proposed task model structure. The basic idea of our approach is to identify typical configurations in task, domain and presentation models and to investigate how they relate each other.

The rest of this paper is organized as follows. In section 2, we will discuss some related work with a focus on the mapping problem. In section 3, we will briefly discuss some application-domain concepts in order to investigate the relation between application functions, tasks and domain objects. Then we will describe our task modelling framework and discuss its benefits for user interface design. The paper ends with conclusion in section 5.

II. RELATED WORK

The mapping problem has been defined in [10] as a key problem for the gradual transformation of models from abstract to concrete level as well as for the mapping 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 [4] as well as development of a description language and tools supporting the specification transitions [6].

A problem with the mapping between models is that they require a detailed representation of model components along the development process. This means that we need to consider at least three aspects when exploring the mapping space: (a) the model type (e.g. domain, task, presentation, and platform); (b) the hierarchical structure of each model (e.g. dialog unit, interaction object), and (c) the progression level along the development life cycle. Up to now, only interface models have defined transitions from abstract to final representations of the design.

TABLE 1. Task categories in CTTE

Abstract	Interaction	Application	User	Cooperative
				

The last two dimensions seem to be most neglected in existing approaches. The problems are mainly located in task and domain models since it is not clear how they relate each-

Manuscript received March 31, 2005. This work was supported in part by the PN 202/2003 national research project) and the FP6-507609 european research project (SIMILAR).

Costin Pribeanu is with the National Institute for Research and Development in Informatics – ICI Bucureşti, Bd. Mareşal Averescu No. 8-10, 011455 Bucharest, Romania (phone: +40-(0)21 2240736; +40-(0)21 2241030; e-mail: pribeanu@ici.ro).

other and which are the transitions they have to undergo in order to effectively support the design process.

Another important problem in model-based approaches is the tools needed to assist designers in building models and handling mappings. The Concur Task Tree Notation (CTT) has been implemented in the CTT Environment (CTTE) [7] which is providing with a graphical notation for task representation (see Table 1) and temporal operators (see Table 2).

TABLE 2. Temporal operators in CTTE

Binary operators	
Choice	T1 [ ] T2
Order independecy	T1 |=| T2
Interleaving	T1 ||| T2
Synchronization	T1 |[ ]| T2
Enabling	T1 >> T2
Enabling with info passing	T1 [ ]>> T2
Disabling	T1 [> T2
Suspend / resume	T1 |> T2
Unary operators	
Optional	[T1]
Iteration	T1*

There are some restrictions in combining binary and unary operators.

This notation has been integrated in Teresa [8], a task-based design tool offering facilities for the computer-aided design of UIs.

III. THE APPLICATION DOMAIN MODEL

A. *Application functions*

The application domain model also referred to as domain model or concepts model defines the concepts related to the domain of the target application. Two components are important: the application functions and data model. Each function corresponds to a business goal which is accomplished by carrying on one or several user tasks.

This mapping is often neglected in the development of interactive systems, although it provides with an important bridge between software engineering and human-computer interaction. In order to present our work we will use the example. The purpose of the application is the management of data about products, clients and orders in a trade company. The mapping could be expressed as a task model represented in Figure 1 by using the CTT graphical notation.

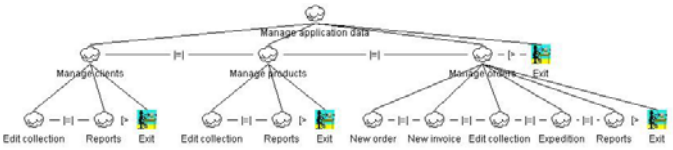

Figure 1. Mapping of application functions onto user tasks

Only high level tasks are represented in Figure 1. This mapping is producing the initial representation of tasks which are relevant for the target application. For the sake of simplicity, we will focus on the task "Record order" represented in Fig. 3. The task goal is to record new orders submitted by clients.

B. *The Data model*

The data model is capturing representations of domain objects (entities), relationships between domain objects and domain object attributes. The conceptual data model is depicted in Figure 2 according to an entity-relationship-attribute (ERA) formalism produced by the DB-MAIN tool [3]. This formalism has been chosen for its expressiveness in representing relations and its capabilities to output both a SQL and XML specifications of the model.

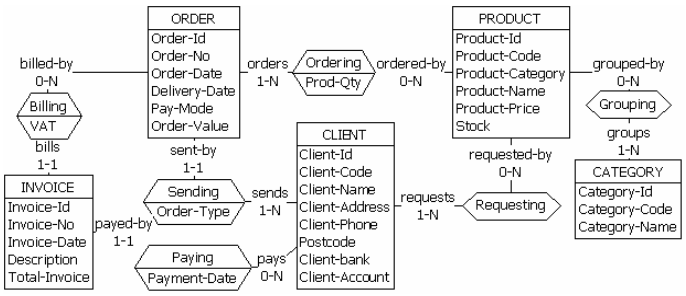

Figure 2. The conceptual data model

As pointed out by Thevenin and Coutaz [11], the domain concepts could be ranked upon the the degree of importance for the application domain and the centrality for the user task. For example, the order, client and product are central entities for the task „Record orders".

In turn, the centrality of objects is determining the centrality of relationships and roles. Central relationships are the relations a central object has with other objects in a task. In order to have a complete mental model, it is important for the user to understand the relationship between domain objects. In this respect, the user should perceive all products which have been specified in an order (perceiving the 1-N relationship) in order to have a feedback on task completion. In a similar vein, it is useful to visualize the client data (perceiving of the 1-1 relationship) in order to be sure that a customer has been specified.

A binary relation has two roles and each role has its own cardinality expressed as an interval (e.g. 0-1 or 1-N). The relevant cardinality is different and depends on the centrality of the object responsible for that role. For example, the relevant role in the relation « Sending » is « sent-by », denoting that an order is sent by only one client. Similarly, in the « Ordering » relation the relevant role is « orders » with the cardinality 1: N, meaning that several products are requested by an order.

As it was pointed out in [9], cardinality of roles is important since it reveals repetitive tasks and helps in task decomposition. Two examples are represented in Figure 4:

- The task „Record order", which is explicitly, started by the task "Take order" (in a work session we might have no order).
- The task "Product data", which is implicitly started (an

order should specify at least one product).

A task enables the user to perform operations on domain objects. The operations may be performed at collection level (create new objects, delete or associate existing objects) or object level (modify the object attributes). The data model elements are selected according to task goals. In this respect, tasks are filtering data model elements which are further needed for user interface design.

IV. TASK MODELLING

A. Layers in task decomposition

As pointed out by activity theory [5], tasks are goal driven, being performed consciously, while actions are depending on operational conditions of the task and become automatic by practice. This distinction is important since it reveals two layers in the task structure: a functional (planning) layer that does not depend on the target platform and an operational layer. However, the operational conditions are a concept which has been further refined for UI purposes [9] by addressing two kinds of variation:

- Variations in using an interaction object: different actions required to manipulate the same interaction object. In this case, the activity is driven by requirements to adjust the work, like articulator actions (e.g. scrolling a list).
- Variations in the platform used to carry on a task: the same goal could be achieved using different user interfaces. This situation corresponds to tasks that are driven by operational goals in a given context of use.

Therefore we will consider goals at both levels of task decomposition, but with different relevance.

The first layer in the task model is represented in Figure 3 and shows how users are planning task performance by decomposing a task in sub-tasks and giving an ordering preference for each of them.

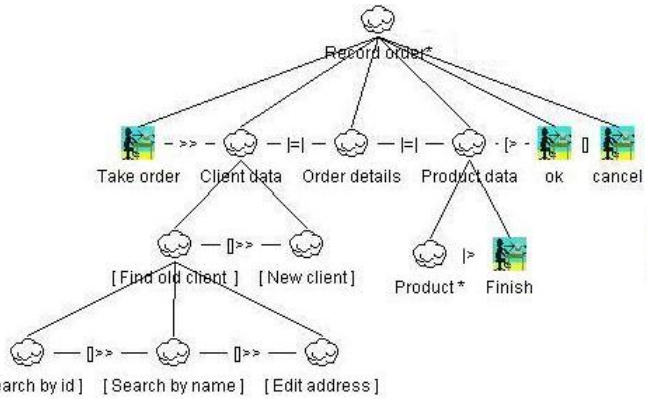

Figure 3. Decomposition for the task "Record order"

Usually, this specification results from early task analysis and represents "what-to-do" knowledge. In our design framework, this layer is the result of decomposition of the functional tasks which are represented in Figure 1. The decomposition stops at unit task level, defined by Card, Moran & Newell [1] as tasks the user really wants to perform.

The second layer represents the operational structure of unit tasks. The decomposition of a unit task stops at basic tasks level, which has been defined as the lowest task level that is using a single interaction object or a single external object or serves a communicational goal [9].

Basic tasks are interaction driven and represent the "how-to-do-it" knowledge, since they show how a unit task will be actually carried on, by using various interaction techniques. According to the interaction object type, we distinguish between information control and function control basic tasks.

Task modeling is an iterative process of identification and description of tasks. Several criteria could be applied for task decomposition but their relevance is varying according to the task model layer:

- Functional: tasks associated with the same business goal. This criterion applies for task decomposition at functional level for the mapping of application functions to user tasks.
- Semantic: task performing an operation onto the same domain object. This criterion is applied to separate tasks which refer to the same object or to the same operation (add new, delete) or to the same interaction method (when several methods are available to accomplish a goal). It is relevant for both functional and operational level and helps in the identification of unit tasks.
- Task object: tasks performing operations with the same interaction object or external object. The criterion is relevant for the operational level and helps in the identification of basic tasks.
- User and work: tasks are performed by the same user (playing a given role) and are denoting a similar work (manual, interactive, communication). The criterion is mainly relevant for cooperative tasks.
- Temporal: tasks denoting specific temporal constraints (like repetitive or optional performance). The criterion is relevant for the representation of temporal constraints among tasks.

B. Decomposition at operational level

The task model is used as a mediator for the mapping between domain and presentation models. An information control basic task is carried on by using an information control abstract interaction object (AIO) in order to access attribute data. Available commands on the target platform are mapped onto function control basic tasks in the task model and abstract interaction objects in the presentation model.

The operational structure for the unit task „Product data" is presented in Figure 4 and consists of two function control basic tasks (ok and cancel) and seven information control basic tasks. In the CTT notation, interaction objects used for data entry are represented as interaction tasks while those that only display information on the screen are represented as application tasks.

A problem with the decomposition at operational level is related to the definition of unit tasks. Unit tasks have a given

relevance for the user and do not depend on a given platform. However, it is possible for a unit task to invoke the execution of another unit tasks. For example, selecting a product from a category is a task the user really wants to perform since it makes the product specification easier. On another hand, finding a product is also a unit task having its own operational structure, like shown in Figure 4.

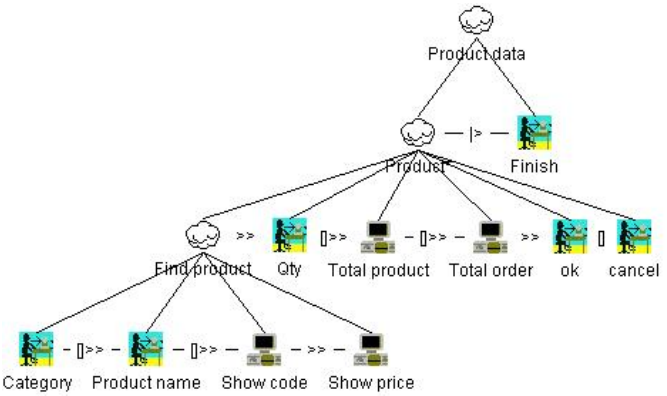

Figure 4. Operational structure for the unit task "Product data"

The operational task model suggests a first level grouping of AIOs. AIO groups are providing with a first level of structuring the interface. As such, they are serving as basic building blocks for the user interface design in a task-based approach.

AIO configurations are the lowest level units of the interface that have associated both a unit task goal and a temporal relation (the ordering of basic tasks).

V. CONCLUSION AND FUTURE WORK

A shortcoming of many approaches is that they start with the data model and not with the task model. As we shown in this paper, the task model is filtering for relevance the data model elements which are further needed in user interface design.

In this paper we presented a task modeling approach for user interface design. We identified three decomposition levels which are relevant in task modeling for user interface design:

- A functional level that results from mapping application functions onto user tasks.
- A unit task level that results from the decomposition of functional tasks regardless the constraints imposed by a target hardware and software platform.
- A basic task level that results from the decomposition of unit tasks which are carried on by using interaction techniques available on a target platform.

VI. REFERENCES

[1] Card, S. K., Moran, T. P. and Newell, A.: *The psychology of human-computer interaction*. Lawrence Erlbaum Associates. (1993).

[2] Clerckx, T., Luyten, K. & Coninx, C.: The mapping problem back and forths: Customizing dynamic models while preserving consistency. In Palanque Slavic and Vinckler (Eds), *Proceedings of Tamodia 2004.* (2004) 99-104.

[3] Englebert, V., Hainaut, J.-L.: GRASYLA: Modelling case tools GUIs in metacases. *Proceedings of CADUI 1999* (Louvain-la-Neuve, 21-23 October). Kluwer Academics, Dordrecht (1999) 217–244.

[4] 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

[5] Leont'ev, A.N., *Activity, consciousness and personality*, Englewood Cliffs, Prentice Hall, (1978).

[6] Limbourg, Q. & Vanderdonckt, J.: Addressing the mapping problem in user interface design with USIXML. In Palanque Slavic and Vinckler (Eds), *Proceedings of Tamodia 2004* (2004) 155-164.

[7] 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

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

[9] Pribeanu, C. & Vanderdonckt, J.: Exploring design heuristics for user interface derivation from task and domain models. *Proceedings of CADUI'2002,* Kluwer Academics, Dordrecht (2002) 103-110.

[10] 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.

[11] Thevenin, D. & Coutaz, J. : Plasticity of User Interfaces: Framework and Research Agenda. *Proceedings of INTERACT'99*, IOS Press Amsterdam, (1999.