TOWARDS A PATTERN LANGUAGE FOR UID

Costin Pribeanu
ICI Bucharest, Romania
pribeanu@acm.org

Abstract
A challenge for the model-based design of user interfaces is to understand how usable designs could be specified as configurations of smaller parts along a pattern hierarchy. Each configuration of these building blocks has to balance requirements coming from several models, which are constraining the design process. This paper aims at investigating typical interaction structures by exploiting the information coming from both task and domain models and to elaborate on a pattern language that captures the essential mappings between these models and the user interface.

Keywords
task-based design, user interface design, operational task modeling, relationships in domain models

Introduction
In his seminal book [1], Christopher Alexander advocated for a deeper understanding of the design process. A pattern is a three-part rule expressing a relation between a certain context, a problem and a solution. The solution is a configuration that allows the forces acting in that context to resolve themselves. In his design philosophy, a pattern is itself a pattern of relationships between other patterns and it is both a thing and the way it is generated. The pattern language elaborated by him and his colleagues [2] goes down from towns to urban spaces, houses and building elements. Pattern languages in user interface design are a challenging topic. Some approaches feature a conceptual view on patterns by pleading for the use of patterns as mental models for designers and lingua franca for communication within the design team and between them and clients or users.

User interface design could be seen as a progressive derivation of the interface components from representations expressing relations between users, tasks, domain, and technology. The generating power of a pattern language relies mainly on the patterns of mappings between these models.

This paper aims at investigating how pattern languages could be used as a

foundation for the model-based design of user interfaces. The basic idea of our approach is to identify typical interaction structures in task and domain models and to elaborate on a pattern language that captures the essential mappings between the various models that are used in the development process

Related work

Welie and Veer [20] argued that now, when substantial bodies of patterns have been published, it is time to develop approaches for structuring pattern languages. They proposed a pattern classification for e-commerce applications based on functional aspects and a pattern language structure which follows a top-down decomposition, along a scale of problems: posture type, experience, task and action. An interesting feature of this approach is the layer of experience patterns that are related to a higher-level goal or motivation.

In a similar vein, Molina et al [6] are arguing for using conceptual patterns during the requirements specification and propagating them throughout the next development stages. They are focusing on what is to be done (the problem space) and not on how to do it (the solution space). Conceptual patterns like filters, master-detail forms or predefined selections could be identified early in the development process, during the discussion with the client. Nilsson's work [8] discusses the limitation of models along different platform and the cost-benefit of their portability. In order to encompass model imitations, he proposed the study of mappings between conceptual user interface components.

The conceptual perspective on patterns provides with valuable insights on the role of patterns in user interface design but leaves outside many concrete problems of user interface design and specification. Seffah and Forbig [14] proposed a layering of task models from the most general to the most context dependent as a foundation for a "multiple user interfaces" paradigm in model-based design. They distinguished between horizontal usability (across platforms) and vertical usability (platform specific). Other approaches attempting to refine task modelling for different contexts of use are described in [12, 16]. Like the previous one, they are mainly addressing task-modelling problems with little or no relation to domain and presentation models.

Traetteberg [19] argued for using patterns as reusable model fragments. He proposed several patterns such as item selection, browse aggregation or select from favourites, which are focusing on the mapping between the domain and presentation models. The work of Sinnig et al [15] is also discussing user interface patterns in the framework of model-based engineering. Like in [13], in their approach a distinction between two categories of tasks is made: goal oriented (what-to-do) and feature oriented (how-to-do-it). Task patterns apply for the former category and feature patterns for the second.

Domain-task-presentation mappings

From the perspective user interfaces design, a pattern language could be an attractive approach to deal with the complexity and diversity of user interfaces, which is difficult to manage using existing model-based approaches. Design of user interfaces is constrained by several models: user model, task model, application-domain model and platform model.

Like the patterns of Alexander, UID patterns could help in deriving characteristics of the presentation, which can afford certain cognitive behavior. Using the information contained in the domain model helps in deriving *what* information the user needs to manipulate in order to perform his task. Using the task model makes it also possible to derive *how* to present and organize this information in a usable way and the *ordering* of task performance.

Our approach is based on the domain-task-interface mappings. Figure 2 is illustrating various kinds of patterns that occur in these models and the derivation process as a *horizontal* mapping. Nevertheless, there is also a *vertical* mapping since patterns have a hierarchical structure within each model. In the domain model, two (rarely more) objects are participating in a relationship and each object has several attributes. The task model is a tree having as leaves the basic tasks. The user interface is structured into dialog units featuring various AIO configurations.

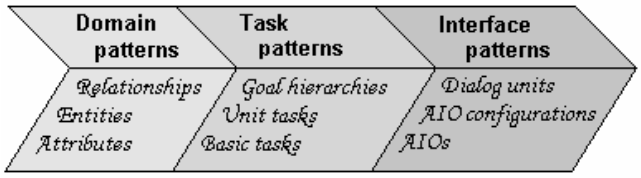

Fig. 1. **A pattern-based framework for domain-task-interface mappings**

We already outlined some typical mappings in the previous section, based on the domain and task models. A simple pattern, which could be observed in the task model, is the *edit entity attributes* pattern. This pattern is exploited in most model-based approaches and is suitable for automatic generation. The task pattern is featuring a sequence of basic tasks starting with a command selection, followed by data entry tasks (corresponding to object attributes) and ended by a confirmation command (usually ok vs. cancel). The pattern applies for editing attributes of both existing and newly created objects. We can identify three occurrences in Figure 1: edit guideline, new section and new base.

The generic form of this pattern is given in Figure 3, where we used the CTT notation for the task model and a wire frame representation for the interface. Most characteristic are the mappings between the three pairs: entity-attributes, unit task -

basic tasks and dialog unit – abstract interaction objects. Attributes in the data model, basic tasks in the task model and AIOs in the interface model are the basic elements in our framework. Depending on the available screen space and the allocation strategy, AIO configurations could be placed into separate dialog units or not.

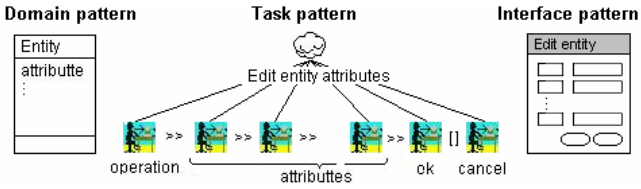

Fig. 2. **A domain-task-presentation mapping for data entry**

Mappings from one model to another are done according to design heuristics and derivation rules. For example, ergonomic rules are used to choose the most appropriate AIOs according to the information coming from the domain model as regarding data type, data length, domain of values or selection type. On another hand, unit tasks names are usually used to label AIO groupings in order to provide with more user guidance.

Presentation patterns

In order to perceive the relationships between domain objects the user needs some additional information to be displayed: either the "one" part (the higher level entity), either the "many" part, either both. We can distinguish between 3 types of such displaying patterns:

a) Showing the higher level entity, for example to display the section to which a guideline belongs – this is usually accomplished by using a text box placed on the top of the AIO group presenting the attributes of the entity;
b) Showing the lower level entities, for example to display the more specific guidelines (recursive aggregation) – this is usually accomplished by using a list box placed at the bottom. This is also used to show associated entities since in a relational model the many-to-many relationship is mapped onto two one-to-many relationships.
c) Showing both the higher and lower level entities, for example to display the general guideline and the more specific guidelines – this could be accomplished using a text box and a list but also embedded dialog units showing a master-detail relationship.

The diagrams in Figure 4 show these typical situations. Although not illustrated, these dialog units also contain the *edit entity attributes* pattern described in the previous section. For example, the pattern in Figure 4c is a composition of three

patterns: *show higher*, *edit entity attributes*, and *show lower*. Typical for these patterns is the placement of the higher / lower level entity above / below the AIO group used for data entry. This visual structure is actually mirroring the "one-to-many" relationship thus being consistent with the mental model of the user as regarding the data organization.

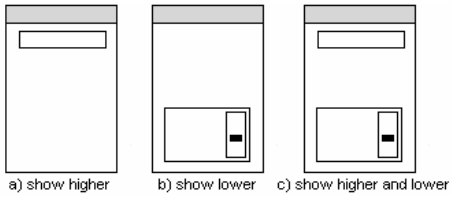

Fig. 3. **Presentation patterns**

The pattern in Figure 4b is applied in master-detail forms where two related tables are displayed. Usually, the lower level entities are displayed as rows of attributes and the resulting pattern is a combination of an AIO group (placed above) corresponding to the "one" part and a data sheet (placed below) corresponding to the "many" part of the relationship.

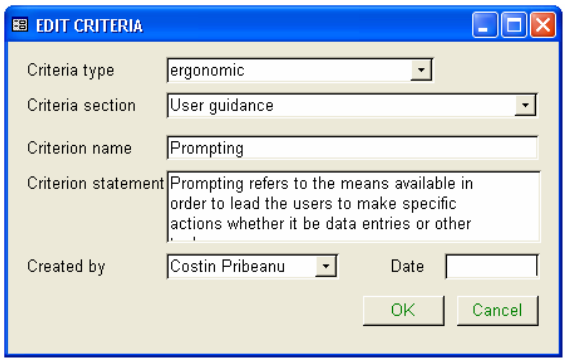

Fig. 4. **An example of presentation**

An example is given in Figure 4. The task is to edit the attributes of an ergonomic criteria in a guidelines management system. The pattern language is composed of following patterns: *show higher, select and show related, show higher, select and show related, edit entity attributes.*

Conclusion

In this paper, we investigated presentation patterns, which are based on the information provided by task and domain models and we have shown how they

could be progressively transformed into design patterns. Display patterns refer mainly to the presentation parts, thus being static. Selection patterns are relating the presentation and dialog model at widget level with the control model while patterns for changing the associations are also relating the dialog model at dialog unit level. The main feature of this approach is that patterns are combining themselves and it is possible to specify large configurations of user interface parts with few patterns.

In order to have a generative power, a pattern language for user interfaces should include both the patterns occurring within the various models and the patterns of mappings.

References

1. Alexander, C. (1979) The Timeless Way of Building. New York: Oxford University Press.
2. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, I. Fiksdahl-King and S. Angel, *A Pattern Language*. New York: Oxford University Press, 1977.
3. Limbourg, Q., Vanderdonckt, J. & Souchon, N. "The Task-Dialog And Task-Presentation Mapping Problem: Some Preliminary Results". In F.Paterno & P.Palanque (Eds.) *Proceedings of DSV-IS'2000*, Limerick, 5-6 June, LNCS 1946, Springer, 2000. 227-246.
4. Molina, P.J., Melia, S. & Pastor, O. "User Interface Conceptual Patterns". In Forbig et al. (Eds.) *Proceedings of DSV-IS 2002*, Springer, 2002. 159-172
5. Molina, P. & Traetteberg, H. "Analyis & Design of model-based User Interfaces". *Pre-proceedings of CADUI'2004*. 211-222.
6. Nilsson, E. "Combining Compound Conceptual User Interface Components". In Forbig et al. (Eds.) *Proceedings of DSV-IS 2002*, Springer, 2002. 114-117
7. Paternò, F. *Model-based design and evaluation of interactive applications.* Springer, 1999.
8. Pisano,A., Shirota, Y. & Iizawa, A. "Automatic generation of graphical user interfaces for interactive database applications". Proceedings of CIKM '93. ACM Press. .344-355.
9. Pribeanu, C., Limbourg, Q. & Vanderdonckt, J. "Task Modelling for Context-Sensitive User Interfaces." C. Johnson (Ed.): *Proceedings of DSV-IS 2001*, Springer, 2001. 167-182
10. Pribeanu, C. & J. Vanderdonckt (2002) "Exploring Design Heuristics for User Interface Derivation from Task and Domain Models". *Proceedings of CADUI'2002*, Kluwer, 103-110.
11. Seffah, A. & Forbig, P. "Multiple User Interfaces: Towards A Task-Driven And Patterns-Oriented Desigm Model". In Forbig et al. (Eds.) *Proceedings of DSV-IS 2002*, Springer, 2002. 118-132
12. Sinnig, D., Gaffar, A., Reichart, D., Forbig, P. 7 Seffah, A. "Patterns In Model-Based Engineering". *Pre-proceedings of CADUI'2004*. 197-210