A Transformational Approach for Pattern-based Design of User Interfaces

Costin Pribeanu[1], Jean Vanderdonckt[2]
[1]*Nat. Institute for Research & Development in Informatics (ICI),*
Bd.Mareşal Averescu Nr.8-10, 011455 Bucharest, Romania
[2]*Université catholique de Louvain, Louvain School of Management*
Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
pribeanu@ici.ro, jean.vanderdonckt@uclouvain.be

Abstract

A pattern-based approach to user interface development is presented that is explicitly based on artifacts contained in a task and a domain models. Exploiting a task model or a domain model in isolation may lead to patterns that are not user-centered. By combining the exploitation of both models at the same time with priority lead to identifying interaction patterns in a systematic way. Types of relationships in these models help in structuring interaction patterns, that are in turn transformed into design patterns for information systems.

1. Introduction

In terms of a pattern language philosophy [1], domain and task models are the main forces competing for a model-based design of user interfaces (UI). User's demands are also competing with her own cognitive capacity and the constraints imposed by the presentation and dialog parts of the UI [4]. This means that design patterns should address both UI development and usability requirements [10]. Since a pattern language is integrating related patterns we will focus on an application example and we will take the following approach.

1. We will identify interaction patterns based on the relationships in the domain model. In this respect we will analyse the categories of tasks, which are afforded by each type of relationship, and we will look for mappings between the domain model and the operational task structure. This also includes a basic mapping with the presentation since task decomposition at operational level is done after selecting appropriate widgets [3].
2. Then we will propose a set of design patterns that provides with a usable solution for both parts of the user interface: *presentation*: grouping of abstract interaction objects (IOs) [3] and allocation of dialog units; *dialog*: interaction at dialog unit level and interaction object level.
3. Mappings between four models are studied: task and domain models on one hand and the presentation and dialog model on the other hand. The proposed pattern is providing with a solution, which consists in the resulting user interface building block. Additionally we will investigate the extent to which control issues could be included in a pattern definition.
4. We will investigate the relation between design patterns and a method for integrating them into a pattern language. For this purpose, we will further exploit the task and domain models in order to integrate the resulted building blocks into bigger UI components.

2. Finding Design Patterns

2.1 Identifying interaction patterns in one-to-many relationships

In most of the situations when the user wants to perform tasks using hierarchically organised data it is important to provide him with means to visualise the relationship between entities. Depending on the relationship type several interaction patterns could be identified in the domain model. In this respect we can say that relationships are affording certain tasks [9]. These interaction patterns are potential since they reveal inherent capabilities of the interactive system as provided by the domain model. Both domain objects and relationships could be visualised and modified by the user and there are several ways to provide him with usable interaction techniques to do it. On the other hand, these patterns are typical interaction structures which could be identified both in the domain and task models. The task model at operational level is a useful artefact because it shows how a unit tasks is actually decomposed in basic tasks pointing to interaction objects of the interface. Interaction patterns are closely related to unit tasks and they are intended to be usable operational structures. In this respect, complex operational structures having several levels of unit tasks, like the one depicted in Fig. 1, are the target of this research. The general case of the one-to-many relationship is illustrated in Fig. 1. There are two object types involved:

section and guideline. The name and length of each attribute is not relevant for our purpose. However, there is an additional construct – the foreign key that is pointing to the primary table (section). The dialog model at interaction object level manipulates this attribute in various ways to satisfy user's task requirements.

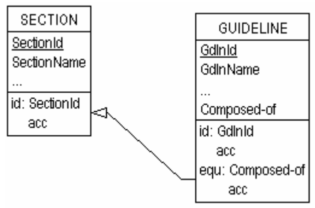


Figure 1. One-to-many relationship.

Another relationship type taking a similar form is classification [13]. For example, guidelines are classified as regarding importance level (good practice, experimentally validated). The mechanism of manipulating the relationship is similar in that a foreign key attribute is used in the first table in order to have access to the data in the second table. To perceive the relationship between domain objects the user needs some additional information to be displayed: either the "one" part (the higher level entity), either the "many" part. We can distinguish between 3 types of displaying patterns, which are supporting tasks afforded by the relationship [5]:

1) Showing the higher level entity, for example to display the section to which it belongs – this is usually accomplished using text box placed on the top of the interaction object group presenting the attributes of the entity;
2) Showing the lower level entities, for example to display the more specific guidelines (recursive aggregation) – this is usually accomplished using a list box placed at the bottom;
3) 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 (Fig. 2).

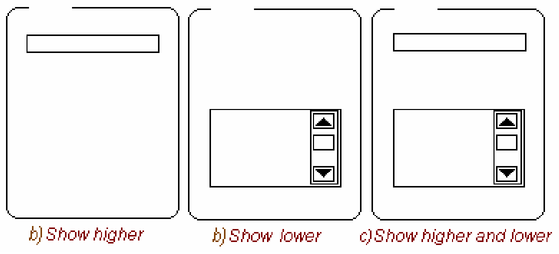


Figure 2. Display patterns in one-to-many relationships.

More complicated interaction structures occur when the task is to change something in the relationship: add or delete an item from a collection or move an item from one collection to another. In this case there are several types of tasks: displaying an item, selecting an item and performing some action upon the selected item. We can distinguish between three interaction patterns:

1) Selecting the higher level entity while editing the attributes of the part entity, for example, selecting the section while editing a guideline (moving a guideline from a section to another i.e. editing the relationship itself) – this could be done using a selecting device (a drop down list).
2) Selecting a lower level entity from a collection in order to perform some action upon it - for selecting a criterion from a criteria group – this could be accomplished with two associated selection devices (selection of higher level object updates the content of the list) and additional function control objects, applying to the selected entity in the list;
3) Selecting an entity from the hierarchy and performing some actions upon, for example a section or one of its guidelines – this could be accomplished with two associated lists (selection of higher level object updates the content of the list) and additional function control objects, applying to the selected entity.

The interaction pattern in Fig. 3b is the best as information provided to the user: he can perceive both the objects and their relationship. This is why it is recommended when performing editing operations on the lower level objects. This interaction pattern corresponds to the situation when the focus is on the management of these objects. The interaction pattern in Fig. 3c could raise some usability problems since there are two lists and in each list an object could be selected. Therefore some additional information denoting the selection to which editing operations apply should be provided. For example, displaying the object type (guideline or section) on the right of buttons. Even so, this pattern should only be designed for experienced users or for special context of use. Depending on the user's task the relationships could be manipulated in several ways thus expanding to many other cases. In each situation, specific interaction objects for function control can be used. For example, in e-commerce applications selection is often combined with the display of related content in a separate window. We could also consider that the user might want to examine in more detail an object from the list. In this case a "show" button could be added or just a double click on the item could invoke a dialog unit where all attributes of the given object are displayed. Also, more complex user interface constructs could be derived after selecting the first object, like for example master-detail dialog units.

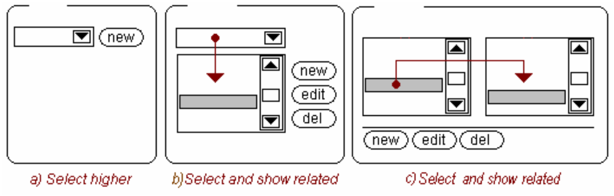


Fig. 3. Interaction patterns in one-to-many relationships

An analysis of relationships between domain objects may be complemented with a task analysis taking into account the tasks the user may want to perform. The domain model itself does not provide with information enough to derive a UI since there is a huge potential of possible tasks.

2.2 Identifying interaction patterns in many-to-many relationships

For this of relationship (Fig. 4), two constructs are resulted from localisation: foreign keys which are pointing to the data in the primary table; a new relation (table Item-Association) that holds the explicit associations. Several tasks could be afforded by this kind of relationship. We distinguish two interaction patterns supporting the specific tasks afforded by a many-to-many relationship:

1) Showing the associated objects, for example showing the criteria respected by a guideline – this goal could be achieved by using a list box with associated entities and one or more functional control objects for editing the relation (add new items or delete existing ones);
2) Changing the current association by removing or adding an existing object to the list – this could be done by using an accumulator (Fig. 5).

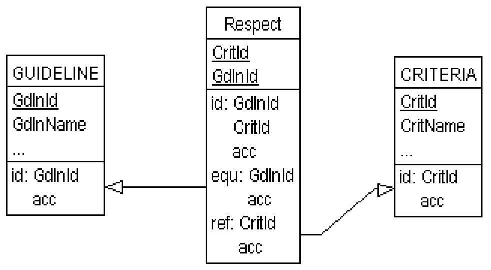


Figure 4. Many-to-many relationship.

In the first case editing the associated objects is not a typical task. This is usually done apart, within the context of their organization. For example, ergonomic criteria are organized in general and elementary criteria. In this case interaction patterns afforded by one-to-many relationships apply. In the second case, a device for selecting several values is used (typical for many-to-many relationships).

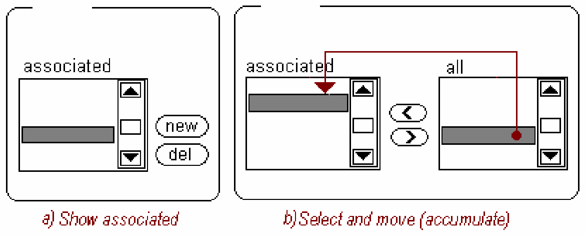


Fig. 5. Inter. patterns in many-to-many relationships.

2.3 Transforming interaction patterns into design patterns

Design patterns derived from the task and domain models. IOs are embodying basic interaction techniques. They are covering both presentation and dialog model at a basic level, which we may term as lexical level. In this respect, interaction objects are the basic constituencies of the user interface. IO groups, which have one or more information, control IO and one function control IO provide with a first level of structuring the interface. Interaction object groups could be used as basic building blocks for the presentation model in a task-based approach. An example is given in Fig. 6 where searching by identification number is used to identify an old client. Grouping of interaction objects is done around the function control.

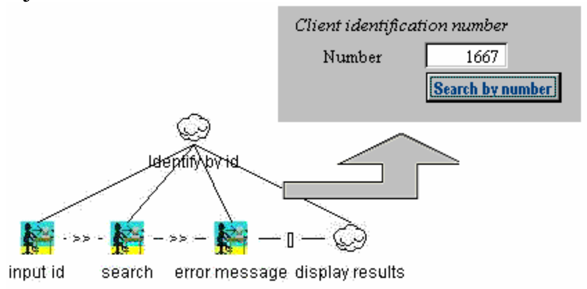


Figure 6. Design pattern for a task-based IO grouping.

We can further decompose the task “display results” with CTTE [7] (Fig. 7) in order to obtain we have the full operational structure for the task “identify-by-id”, including the feedback presented to the user. The task structure suggests a grouping of interaction objects in the interface according to semantic and functional criteria [6]. IO groups could further be grouped together to form higher-level groups. An example is given in Fig. 8 where three groups are grouped following a higher-level goal in the goal hierarchy. According to ergonomic criteria, this is good for 3 reasons: (i) provides for user guidance, by grouping related interaction objects; (ii) reduces memory workload, by creating chunks of information and reducing the articulator distance needed to perform a given task; (iii) provides for compatibility with the user tasks. We can put it all together and propose a design pattern. In order to propose a pattern language we will start by describing the following: problem, context, forces, solutions and comment.

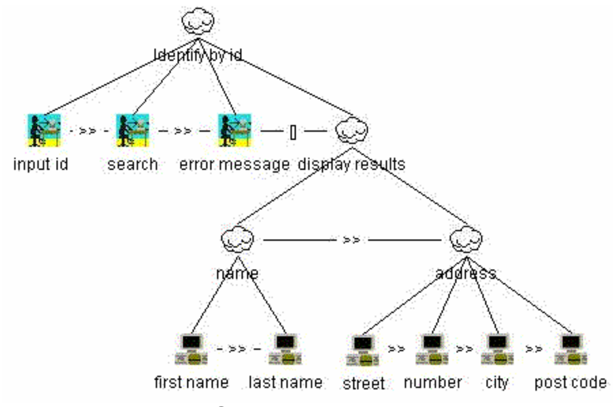
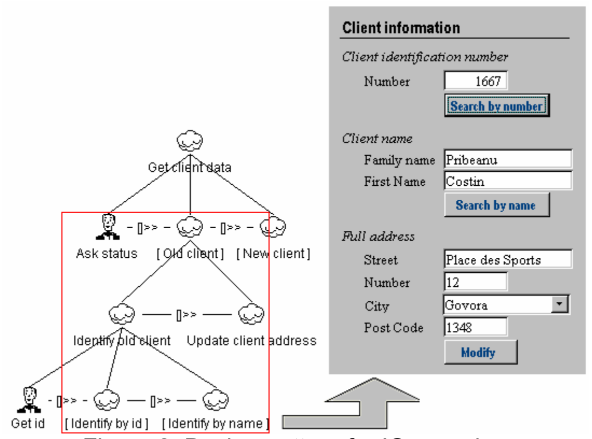


Figure 7. Operational task structure



Client information

Client identification number

Number: 1667

Client name

Family name: Pribeanu

First Name: Costin

Full address

Street: Place des Sports

Number: 12

City: Govora

Post Code: 1348

Figure 8. Design pattern for IO grouping.

We can also derive design patterns based on display patterns like those presented in Fig. 3. We will take the more general example depicted in Figure 3c and we will propose a pattern which is mainly based on one-to-many relationships in the domain model [8].

Problem

Functions could be chained such as the same data is used as prompting for future actions and feedback for previous actions. Ideally, the user should be provided with semantic feedback showing the effect of his actions to the application data.

Context

The same operational structure could be used in several sub-tasks. It usually happens when there are several functions performed upon the same object. For example, the client address is used as feedback after two search methods and for data entry for a new client or when the address changes.

The user is performing a search function. If successful, the attribute data of the found object is displayed in the interface. Then the user could act upon the displayed data in order to perform further actions. This situation is typical in data base applications when we first search for a record and then edit it.

Forces

There are several search keys. For example, the user can search using a client id, a personal id number or the name. Some search methods could be faster other could be more easy to use. If a method fails, possible because of a data entry error, the user might want to try another.

There are a huge number of possible groupings of AIOs in the interface. Grouping of interaction objects could be done according to semantic criteria provided by the data model or in a task based approach. Semantic criteria help to perceive the data structure including relationships. A task-based approach minimizes user actions.

There is always a tradeoff between the information density and the articulator tasks for navigating between different dialog units.

Solution

First level of AIO grouping should mirror the operational task structure. Assign a static interaction object denoting the semantics of data or function to each AIO. This design step is performed in a bottom-up approach.

Higher-level groups are based on the goal hierarchy. Assign a static interaction object denoting the task goal to each AIO group. Use up to three levels of grouping in a dialog unit. Allocation of dialog units should be done in a top-down approach based on the task model.

Comment

This pattern applies mainly for the presentation part of the interface and helps in organizing the information on the screen in a way that provides with user guidance. It can be integrated in more complex patterns.

An example using this pattern is presented in Fig. 12 where a dialog unit for the task "edit guideline" was designed. The user can manipulate one-to many relationships as follows. Base name is only displayed. Section could be selected from a drop-down list. If not found, a new section could be created. The user could also select the general guideline and he is provided with a list of more specific guidelines. A classification relationship is used to select the importance level of the guideline from a drop-down list. In this example, the recursive aggregation of guidelines was implemented according to the case 3c in order to provide with maximum of feedback. Seeing both the ancestor and the descendants helps the user to better perceive the underlying domain model. However, the design decision is also dependent on the available screen space. Design patterns derived from interaction patterns. More complicated interaction structures like those afforded by many-to-many relationships in the domain model need a closer look in order to be embedded in useful task structures and thus aid the UI design.

Task goal	Presentation	Dialog	Control
select a category		click to show list + select + show current	category id – internally available category name – displayed
add a new category	new	Click on button	
edit attribute value	base id	Enter	text box value – internally available
record / cancel	ok cancel	Click on "ok"	perform transaction (record AIO values)
			update the category list
		Click on "cancel"	

Table 1: Interaction techniques for creating a new entity

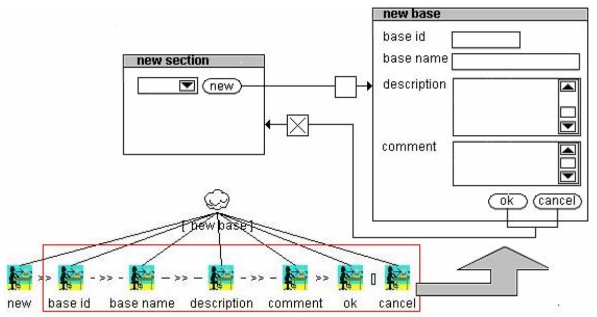


Figure 9. Design pattern for creating a new entity.

For example, the operational structure of the unit task "new base" (Fig. 1) suggests a dialog unit having one group composed of interaction objects for information control and a group composed of two interaction objects for function control. The interaction pattern is simple: the user is pressing the "new" button while editing the section. Then the dialog unit is displayed so the user can enter attribute data by using information control objects like text boxes (profiled text box for base id and base name and multiline text boxes for description and comments. Then she can record the new base in the database by pressing the "ok" button or cancel the operation by pressing the "cancel" button. The task and domain models are providing with useful information for user guidance: the dialog unit will display the name of the unit task in the window title; static interaction objects (labels) denoting the semantics of data are composed with the information control objects (Fig. 9). Both presentation and dialog part at dialog unit level are provided. This pattern is composed from several abstract interaction objects (generic interaction techniques), which are, providing with presentation, dialog and control at interaction object level. These patterns are presented in Table 1. Although this design pattern is well known, it is mentioned because is needed when combining several patterns in a pattern language. It also shows how we van integrate useful information from both task and domain models. In this respect, showing the current selected value for the base and the possibility of adding a new base corresponds to the interaction pattern in Fig. 4b. This pattern occurs three times in the task model: when selecting the section for a guideline; when selecting the base for a new section; when selecting a criteria section. In the third case we will not provide a new button since the set of criteria is well-established. This is a variation in the context of use, which should be recorded in the context and solution parts of the pattern. In this respect, the pattern language should be flexible enough to capture different situations of use if they lead to a similar design. Since a strong feature of patterns is to be context sensitive we will need to describe several solutions: a general one, corresponding to the common part of the problem and several detailed solutions, corresponding to the diversity of the use situations.

This pattern is based on the display pattern (Fig. 3a) and the interaction pattern depicted (Fig. 4a). It includes the design pattern described in the previous section. Additionally, it describes the whole interaction process by including the data input for the higher-level object and the relation between the two dialog units. From an implementation point of view, this data is recorded in a transaction process when the user is pressing the "OK" button (explicit user action). In this case, the attributes are taking the values stored by the interaction objects and the drop down list is updated with the new value. Ideally, the new category should be displayed as the current selected value in order to save the selecting action.

We will further analyze the two situations described in Fig. 6, but in a more concrete task context. For example, the relation between guidelines and criteria could be edited in a separate dialog unit, where both unit tasks of adding a new item and deleting an existing one are possible. The diagram in Figure 10 shows the mapping between the task and the presentation models. In Table 2 the mappings at interaction object level are presented. A different context for editing the association is presented in Fig. 11. In this case the user is provided with a list of associated items. He can remove an association or she can add a new one by se-

Task goal	Presentation	Dialog	Control
Main dialog unit			
see associated	label+ button	Click on button	displays the dialog unit
Relationship dialog unit			
select an object	any list	scroll+click on item	object id internally available
associate	list box “all”	click on button ”<”	creates a record in the relationship table
delete an object	list box “associated”	click on button ”>”	deletes a record in the relationship table
return to main	button “close”	Click on “close”	close the relationship windows

Table 2: Interaction techniques for editing associations.

Task goal	Presentation	Dialog	Control
Main dialog unit			
select an object	list box “associated”	scroll+click on item	object id internally available
delete an object	list box + button	click on button ”X”	deletes a record in the relationship table
associate	list box + button	click on button ”+”	opens the “associate” window
Available objects dialog unit			
select a category	category combo box	click + select	updates “all” list box
select an object	list box “all”	scroll+click on item	object id internally available
associate	list box	click on button ”<”	creates a record in the relationship table
return to main	button “close”	Click on “close”	close the relationship windows

Table 3. Interaction techniques for an association.

lecting it from a list. This is a better solution since the user is provided with some information about the existing state of association. This requires more screen space in the first dialog unit.

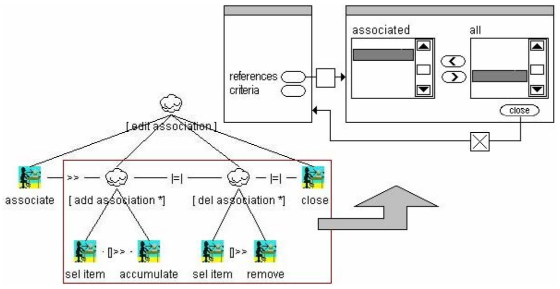


Figure 10. Editing a many-to-many rel. in one dialog unit.

A design pattern integrated both situations is presented below. The interaction techniques are presented Table 3. The item collection could also be structured. In this case the user will first select the category and then the item. Actually this is a combination of association and aggregation relationships. It is possible to create pre-defined (typified) dialog units that accept as parameters the data source extracted from the domain model. This way the code needed to handle the dialog is re-used for similar interaction structures.

The data model could be also exploited in order to derive appropriate interaction objects (for example, criterion instead of item) and to edit their properties (list width). In Fig. 8 an example of presentation derived from task and domain models is presented. It corresponds to the operational task structure in Fig. 1. For each domain object associated with a guideline we have an interaction object group in the UI. Each group has a label, a list box and two buttons.

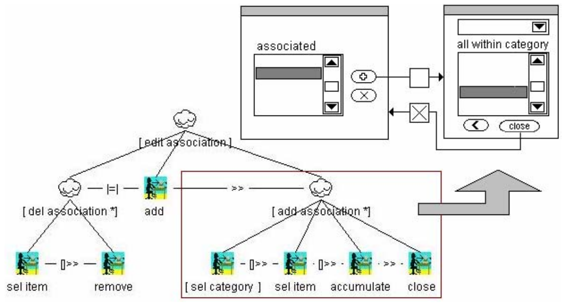
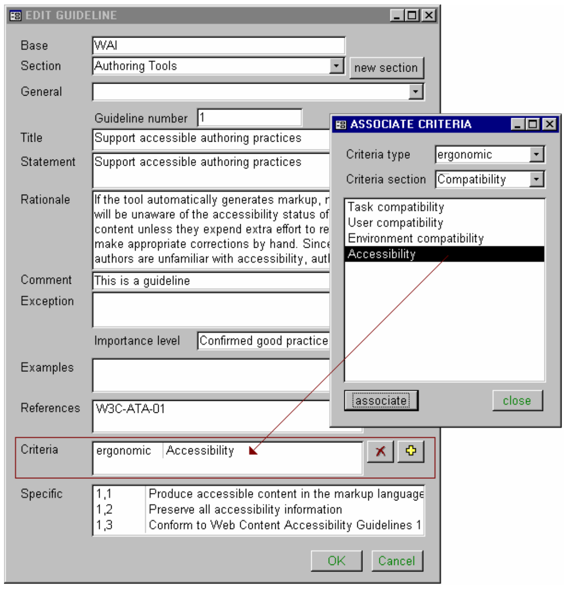


Figure 11. A many-to-many rel. in two dialog units.


EDIT GUIDELINE

Base: WAI
Section: Authoring Tools
new section
General:
Guideline number: 1
Title: Support accessible authoring practices
Statement: Support accessible authoring practices
Rationale: If the tool automatically generates markup, r
will be unaware of the accessibility status of
content unless they expend extra effort to re
make appropriate corrections by hand. Since
authors are unfamiliar with accessibility, autl
Comment: This is a guideline
Exception:
Importance level: Confirmed good practice
Examples:
References: W3C-ATA-01
Criteria: ergonomic | Accessibility
Specific:

1,1	Produce accessible content in the markup language
1,2	Preserve all accessibility information
1,3	Conform to Web Content Accessibility Guidelines 1

OK Cancel

ASSOCIATE CRITERIA

Criteria type: ergonomic
Criteria section: Compatibility

Task compatibility
User compatibility
Environment compatibility
Accessibility

associate close

Figure 12. Adding a criterion while editing a guideline.

3 Towards a Pattern Language

A problem with pattern languages for user interface design is the difficulty of relating them. For example, we need a more complete set of patterns in order to include cross-references in each pattern. However, this seems to go very deep in the design process and prone to reduce generality. In this moment it seems to be more feasible to catch only a general design problem and to develop and document a set of related patterns. While the pattern definition could follow a template, there is still a need of some common information to discuss, illustrate and justify the pattern. Since the task of elaborating of patterns is time consuming and may need further steps (in order to gather valuable design knowledge from several designers) we propose a pragmatic approach, based on a short template (formal description) for each pattern and additional (informal) description including examples, discussion, figures and tables.

Together these will form an exploratory framework for identifying new patterns and will preserve the findings for further work. This initial form of pattern language will also serve as a design rationale for the user interface. When enough knowledge is gathered, the template could be expanded in order to record in a formal way the pattern definition. However, a pattern language will integrate patterns with different granularity, according to their scope and scale. Different scope means that patterns will mainly focus on some component: presentation, dialog or control. It also means that they will be different in covering design and implementation.

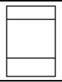
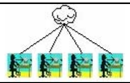
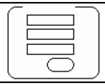
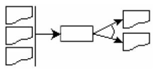
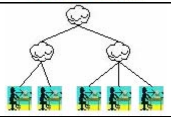
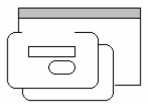
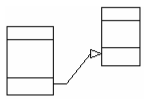
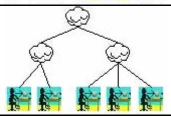
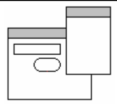
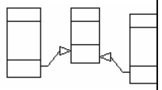
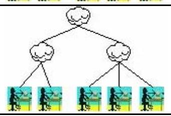
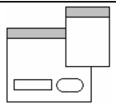
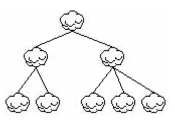
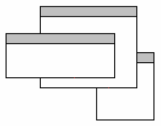
Domain	Task model	Presentation
		
		
		
		
		

Table 4: UI derivation from task, domain models.

Different scale means that only that for large-scale patterns how to combine other patterns will be described. In order to develop a usable pattern language we need to start with some basic heuristics, which are too simple or too informal to be recorded in a complex template but which, are useful in identifying and justifying a pattern. On another hand, having this basic layer, pattern descriptions could be more concise and thus easier to elaborate and manage. A more synthetic presentation of our task-based approach is depicted in Table 4. First row shows how interaction object groups are derived from task and domain models. Domain objects are providing with attribute information from which interaction objects for information control are derived. In our task modelling framework this row corresponds to the operational level in task decomposition and shows how unit tasks are performed with a given technology. Grouping of interaction objects is done either according to the semantics of the domain model (rarely, for objects having many and / or compound attributes) or around function control IOs (often). Static IOs denoting the group meaning are also added in order to increase user guidance. For example, IO groups could take the name of the unit task they support. This mapping is shown in the second row. Third and fourth rows represent the derivation of more complex unit task structures by considering relationships between domain objects. In turn, these structures are further exploited in order to derive interaction object groups and dialog units. The last row is incomplete in that it illustrates allocation of dialog units only by considering the task model. In a task-based approach early task modelling is assumed to integrate systems functions. Therefore they are not further exploited during operational task modelling. Clearly, these mappings are not a substitute for patterns. However, they show an underlying task-based design philosophy and help in organising design patterns in a more systematic way and describing them in a more concise manner.

No.	Statement
H1	Assign a static interaction object, denoting the data meaning, to each information control object.
H2	Assign a static interaction object, denoting the goal name, to each goal at unit task level.
H3	First level grouping of interaction objects should mirror the operational task structure.
H4	Higher level grouping of interaction objects should mirror the goal structure
H5	Assign a static interaction object to each higher level grouping of interaction objects, denoting the goal it represents

Table 6: Heuristics for grouping of interaction objects.

This basic level could be then completed with heuristics for typical derivation rules providing with more detailed design knowledge. For example, in a previous work several heuristics were proposed for grouping of interaction objects (Table 6). We can use heuristics for dialog unit allocation following a given strategy. Heuristics could be illustrated with examples. This basic layer will be the most general in that it will provide designers with heuristics and design rules that apply to a variety of patterns. We can summarize its content as following: ergonomic criteria; selection rules for choosing the most ergonomic IO (interaction technique); task-domain-presentation mappings, detailed heuristics for the derivation of presentation.

4. Conclusion

Operational task structures describe how users are manipulating domain objects. Interaction object groups are formed by grouping information control objects around a function control object that is designed according to requirements coming from the task model. More complex task structures at operational level are afforded by relationships between domain objects. Not only IO groups could be derived but also dialog units. Thus bigger building blocks having both presentation and dialog parts could be derived. A problem with patterns is the diversity of design situations. Although patterns are intended to satisfy requirements related both to complexity and diversity, it is hard to address methodological aspects:

- Applications are very different as regarding the driving model: driven by task and functional requirements, others by complex relationships or by large data structures and other by content.
- Although attractive, patterns could easily trap the designer into futile work which could be saved by using design heuristics;in elaborating a pattern language, finding a stopping criterion is hard.
- Patterns languages are difficult to elaborate: the more formal definition is used, more time is spent to integrate related patterns and this may lead to narrow their applicability.

In this paper we investigated design patterns, which are based on the information provided by task and domain models. On the basis of our previous work and this investigation we proposed a method to develop pattern languages in two steps:

1. Identifying patterns by using information from task and domain models and recording them in an initial pattern language combining formal definition with informal description;
2. Formalisation of the pattern language by expanding the initial pattern template.

5. References

[1] Alexander, C., *The Timeless Way of Building*, Oxford University Press, New York, 1979.

[2] Bayle, E., Bellamy, R., Casaday, G., Erickson, T., Fincher, S., Grinter, B., Gross, B., Lehder, D., Marmolin, H., Potts, C., Skousen, G., and Thomas, J., “Putting It All Together: Towards a Pattern Language for Interaction Design, Summary Report of the CHI’97 Workshop”, *SIGCHI Bulletin*, 30(1), Jan. 1998. Accessible at http://sigchi.org/bulletin/1998.1/erickson.html.

[3] Bodart, F. and Vanderdonckt, J., “On the Problem of Selecting Interaction Objects”, Proc. of BCS Conf. HCI’94 "People and Computers IX" (Glasgow, 23-26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (eds.), Cambridge University Press, Cambridge, 1994, pp. 163-178.

[4] Card, S.K., Moran, T.P., and Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, 1983.

[5] Chaffin, R., Herrmann, D.J., and Winston, M.E., An empirical taxonomy of part-whole relations: Effects of part-whole type on relation identification, *Language and Cognitive processes*, 3(1), 1988, 17-48.

[6] Furtado, E., Furtado, J.J.V., Silva, W.B., Rodrigues, D.W.T., Taddeo, L.S., Limbourg, Q., and Vanderdonckt, J., “An Ontology-Based Method for Universal Design of User Interfaces”, Proc. of Workshop on Multiple User Interfaces over the Internet: Engineering and Applications Trends MUI’2001, A. Seffah, T. Radhakrishnan, G. Canals (eds.), Lille, 10 September 2001. Accessible at http://www.cs.concordia.ca/~faculty/seffah/ihm2001/ papers/furtado.pdf

[7] Paternò, F., *Model-based design and evaluation of interactive applications*, Springer-Verlag, Berlin, 1999.

[8] Pisano, A., Shirota, Y., and Iizawa, A., “Automatic generation of graphical user interfaces for interactive database applications”, Proc. of CIKM’93, ACM Press, New York, 1993, pp. 344-355.

[9] Storey, V., Understanding Semantic Relationships, *VLDB Journal*, 2, 1993, pp. 455-488.

[10] Mahemoff, M.J. and Johnston, L.J., “Principles for a Usability-Oriented Pattern Language”, Proc. of OZCHI’98, IEEE Press, Los Alamitos, 1998, pp. 132-139.

[11] Vanderdonckt, J. and Gillo, X., “Visual Techniques for Traditional and Multimedia Layouts”, Proc. of 2nd ACM Workshop on Advanced Visual Interfaces AVI'94 (Bari, 1-4 June 1994), T. Catarci, M.F. Costabile, S. Levialdi, G. Santucci (eds.), ACM Press, New York, 1994, pp. 95-104.

[12] van Welie, M., van der Veer, G.C., and Eliëns, A., “Patterns as Tools for User Interface Design”, Proc. of TFWWG’2000, J. Vanderdonckt, Ch. Farenc (eds.), Springer-Verlag, London, 2000.

[13] Winston, M.E., Chaffin, R., and Herrmann, D., A taxonomy of part-whole relations, *Cognitive Science*, 11, 1987, pp. 417-444.