Deriving Presentations from Task Models

F.D.Paternò[1], I.M.Breedvelt-Schouten[2], N.M. de Koning[2]

[1]CNUCE-C.N.R., Via S.Maria 34, Pisa, Italy,
f.paterno@cnuce.cnr.it
[2]BAAN Labs, BAAN Company N.V., P.O. Box 250, 6710 BG, Ede, The Netherlands, ibreedvelt@baan.nl, ndkoning@baan.nl

Abstract

Most task-based approaches have been used to analyse and design the dialogue part of interactive applications. There has been less focus on how a task model can be used to derive systematically indications for the design of the presentation of a user interface. This paper presents a solution to this problem which is based on identifying the sets of tasks that are active during the same period of time, and rules that take into account the semantics of the tasks considered and their relationships.

Keywords

Task models, Design of presentations, Systematic methods for supporting design and development

1 INTRODUCTION

The design of the presentation of modern interactive user interfaces is often complex and requires in-depth design knowledge. It is thus important to identify declarative models and inference mechanisms that significantly reduce the demands on the interface developer. Many aspects of design knowledge are domain-independent so that it only needs to be stated once and then can be applied to many different domains.

Task models (Diaper, 1989) have been recognised as an important element in user interface design (Johnson et al., 1993) as they incorporate knowledge of the user's intentions and activities. However most of the work on task-based design has focused on how to support the design of the structure of the dialogue. Less attention has been paid to supporting the presentation design. A few exceptions, such as the declarative presentation structures in Mastermind (Castells et al., 1997), and the layout rules developed by Vanderdonckt and others (1994), have attempted to provide more structured and declarative approaches to the design of presentations. Another interesting type of approach is in (Zhou and Feiner, 1997) where the generation of a presentation is considered as the development of a visual discourse which is developed top-down with the design constraints incrementally specified at each level.

Some work has been developed following a different approach: considering the data to be presented and their semantic features to identify effective presentations (Mackinlay, 1986) (Roth et al., 1994). We believe these are useful contributions though we assume that effective presentations can only be generated after an integrated analysis of data semantics and possible user tasks.

In the TLIM method (Paternò, 1997) a task model can be designed and then a corresponding architectural model produced so that it complies with the same temporal and semantic constraints indicated by the task model. The method is supported by a tool which automatically supports the transformation of a task model into an architectural model. Further, designers can still interactively customise the transformation for specific aspects of the application considered. This method has raised the interest of industrial developers who are tailoring it for the design and development of Enterprise Resource Planning applications (Breedvelt et al., 1997).

In the current version of the method the problem of designing the presentation of the user interface is addressed only after the architectural model has been defined. This may be too late for those designers who want to have a preliminary view of the user interface presentation.

This paper presents a proposal that uses the ConcurTaskTrees notation as a starting-point for expressing task models. These models implicitly contain design knowledge which can be used for the design of the presentation of the user

interface. We thus need a systematic method that can analyse them, derive this knowledge, and use it for the design of the presentation of the application.
We first give an overview of our approach and then we discuss each individual phase. Finally, we give an example of an application, followed by some concluding remarks and areas for future work.

2 OUR APPROACH

Our approach takes a task model expressed using the ConcurTaskTrees notation (Paternò et al., 1997) and indicates how to design the presentation. These indications are generated by a visit of the task tree. This visit is top-down and it identifies the set of tasks that are enabled in the same period of time and thus can be presented at the same time. These sets of tasks are called "activation sets".
Next we take the tasks that are in the same activation set and indicate how to group them in the presentation of the user interface. These indications are derived from the temporal operators among tasks, and the structure of the task model.
For example, tasks that have to communicate information with each other should be placed in close proximity. If there is a choice among tasks to perform then we know that the possible presentation should highlight what the various choices available are. If there is a disabling task then we know that we have a control task, which may be placed in a predefined location, whose purpose is to disable one or more other tasks. We try to identify whether the group of tasks corresponds to some predefined task pattern. Then we apply specific layout policies in order to define the structure of the overall presentation.
Finally for each basic task we identify the related specific presentation using some predefined task-oriented presentation templates which take into account the semantics of the task. Basic tasks are those tasks which are not further decomposed in the task model. Thus they correspond to the leaves of the task tree.
At this level we consider only application tasks (which can be classified into Overview, Comparison, Locate, Grouping of data, Calculate, Application control types of tasks, depending on the type of use expected for the data presented by the application) or interaction tasks (which can be classified into Select, Control, and Edit).
We also consider the properties of the data that have to be presented (data type, cardinality, presentation type) in order to understand the type of presentation that they require.
We decided not to take information from the task model for the design of the presentation by a bottom-up analysis because this may generate a non-consistent

design which would require many modifications later on. In bottom-up approaches designers associate a possible presentation to each basic task and then they have the problem to compose these basic presentations to obtain the overall presentation of the user interface of the application. We soon realised that associating a presentation to a basic task in isolation with respect to the design of the other basic tasks can conduct to low effective presentations which conflict for the type of design choices and generate bad overall presentations whereas with top-down approaches it is possible first to make the overall design decisions and then refine each basic presentation within a common framework.

3 IDENTIFYING ACTIVATION SETS OF TASKS

We can define a task in terms of its subtasks and related operators. For example, the Printing task can be decomposed into the sequential composition of Selecting a file, Selecting parameters, and Print subtasks.
Since the semantics of many temporal relationships depends on the first action (in our case the first subtask) of the task considered we need to introduce two functions:
First, which takes a task and provides a set of subtasks where each of them can be the first one to be accomplished. In the Printing example it would return the Selecting a file subtask. In this example the First function returns only one subtask but there are examples where it can return multiple tasks.
Body, which takes a task and provides all the subtasks that cannot be the first one to be performed. In the Printing example it would return the Selecting parameters and Print subtasks.

With the support of these two functions we can analyse the task tree to identify the set of tasks which are active at the same time. We have defined some rules to identify the Activation sets which depend on the temporal operator we use:

- independent concurrent tasks (tasks composed by the ||| operator) and communicating concurrent tasks (|[]| operator) belong to the same activation set;
- sequential tasks (>> operator), where the task on the left (when it terminates) enables the task on the right, belong to different activation sets;
- choice tasks ([] operator) belong to different activation sets except their first subtasks which all belong to the same activation set;
- when there is a disabling task ([> operator) its first action belongs to all the activation sets associated with the tasks which can be disabled and its body belongs to another activation set.

We introduce our approach with a short example (Figure 1) where we consider a simplified task model for managing files. The abstract tasks (indicated by a cloud icon) are tasks whose performance cannot be allocated uniquely as they have subtasks that, in this case, are user interactions (indicated by a human/computer icon) or application-only tasks (indicated by a computer icon). At the beginning we have an alternative choice among *Editing*, *Printing* and *Deleting* tasks. Their performance can be disabled by the *Close* task. In the case of editing we decompose the task into opening a file followed by the parallel execution of multiple tasks (*Insert*, *Cut* and *Scroll* tasks). The *Printing* task is decomposed into a different structure as we first have the *Select print* activity (which activates a dialogue box for specifying the print parameters), followed in this case by three sequential tasks: *Select printer*, *Select pages*, and *Print*.

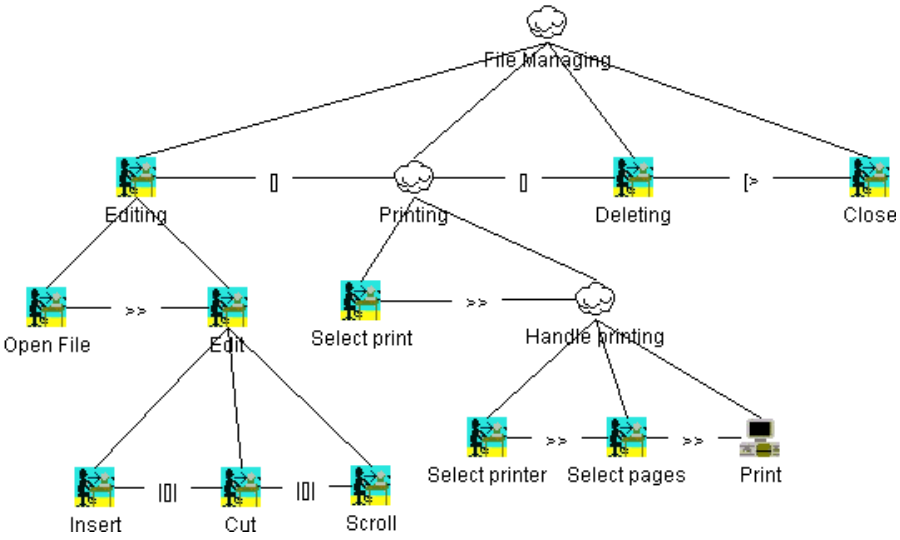

Figure 1: An example of a simplified task model

If we analyse the tree top-down we have to identify the activation sets for each level of the tree. At the first level we find three tasks which are composed by the choice operator ([]) and finally they are composed with another task (by a disabling operator [>). The semantics of the choice operator indicates that at the beginning the first action of each task is available, and as soon as one of them is performed the other tasks are no longer available. On the other hand, the semantics of the disabling operator indicates that the first action of the disabling task is always available and when it occurs it activates the performance of the body of the disabling task, if any. Thus there are five activation sets: one with the

first actions of the tasks, and then one for the body of each task along with the first action of the disabling task and, finally, one with the body of the disabling task.
More precisely, we have:

Activation task sets (Level 1) = {first(Editing), first (Printing), first(Deleting), first(Close)}, {Body(Editing), first(Close)}, {Body(Printing), first(Close)}, {Body(Deleting), first(Close)}, {Body(Close)}.

If we consider the next level the temporal operators that we find do not need any new activation sets. However, we find information that can be used to give more precise definitions of the activation sets identified. For example, we know that the Delete task is considered as a single action task, thus first(Deleting) = Deleting and Body(Deleting) is empty. The same holds for the Close task. Similarly, we know that Body(Editing) = Edit and Body(Printing) = Handle Printing. Thus we obtain:

Activation task sets (Level 2) = {Open file, Select print, Deleting, Close}, {Edit, Close}, {Handle Printing, Close}.
Finally, if we consider the third and last level we see that both Edit and Handle Printing are further decomposed, but Edit is decomposed into subtasks which are active during the same period of time and are thus still part of the same activation set, whereas the subtasks of Handle Printing have to be performed sequentially and thus each needs one activation set. The final definition of the activation sets is therefore:

Activation task sets (Level 3) = {Open file, Select print, Deleting, Close}, {Insert, Cut, Scroll, Close}, {Select printer, Close}, {Select pages, Close}, {Print, Close}.

We can note that one task can belong to multiple activation sets.

In Figure 2 we can see the result of our tool for calculating activation sets according to the rules that we have introduced in this section. The tool has been integrated with the editor of the task models. It is also possible to update the activation sets in case the designer changes the task model and to save the activation sets, and then to activate the second part of the design method which concerns the support for the design of the presentations associated with each activation set.

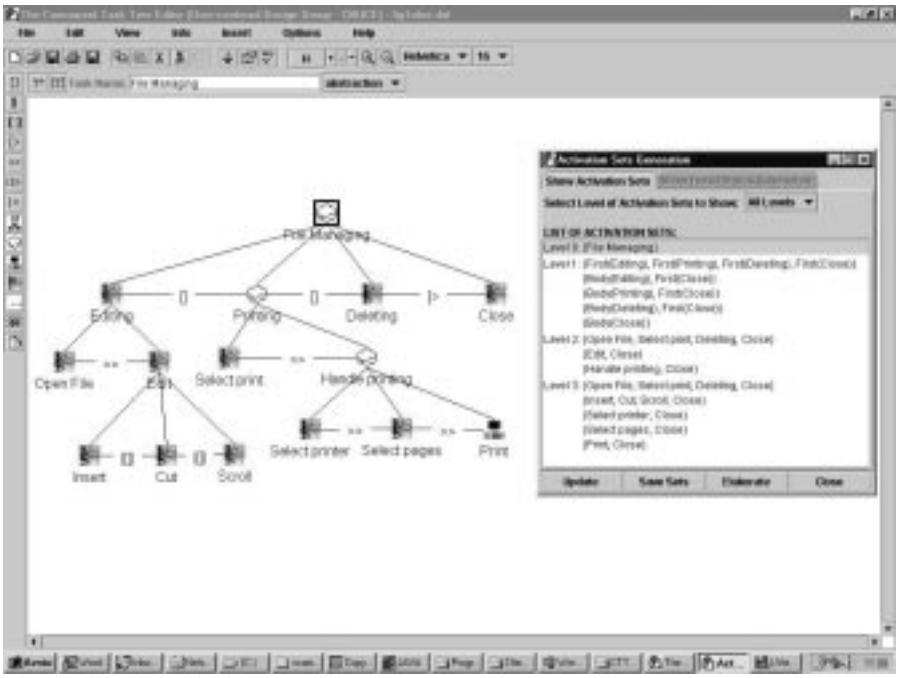

Figure 2: Task model and related activation sets

4 GATHERING INFORMATION FOR PRESENTATION DESIGN FROM THE TASK MODEL

In the activation sets associated with the lowest level of the task tree, only the basic tasks of the task model are involved. The amount of basic tasks per activation set can differ substantially. If there are multiple basic tasks in one activation set we need to determine rules to indicate the most effective structures of presentations taking into account the temporal operators and the structure of the task tree. A *structure for presentation* gives an overall indication of the presentation which can be obtained leaving the definition of some details for the next phase. These structures of presentation can be represented with the support of an automatic tool so that the designer has an idea of the possible impact of the task model on the final presentation.

4.1 Gathering information from the task model structure

An important aspect within an activation set is the structure of the task tree which has generated the set. The tasks which are part of an activation set can be

composed by all the operators except the enabling one. One element is the possibility to identify both groups and subgroups, depending on whether tasks which belong to the same activation set share the same ancestor or parent task. If we consider the task tree at bottom levels, we find that the groups of tasks which share the same parent task are semantically closer to each other than the groups of tasks which share an ancestor at a higher level in the task tree. Figure 3 shows an example of a task model along with the related structure of presentation. The subtasks of the Enter Date task are one group of tightly related tasks. One level higher, the Show Calendar task and the Enter Date task are also grouped in close proximity, since they share the same parent. Again one level higher, the Enter Calendar task and the Enter Project Team task are grouped together.

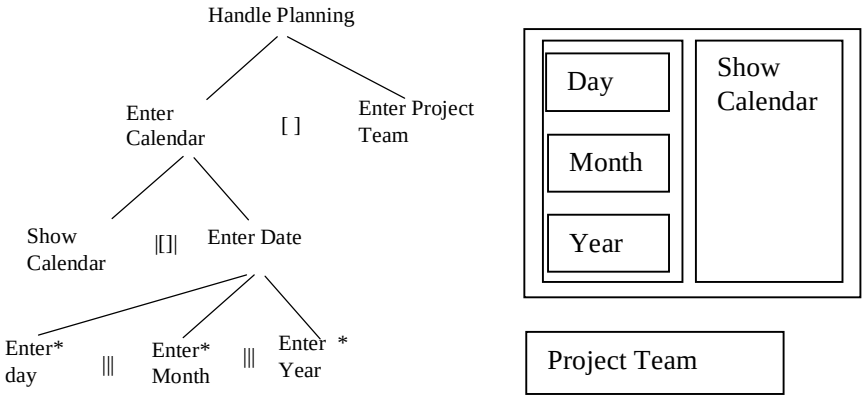

Figure 3: Groups and Subgroups in a Task Model and the corresponding presentation structure

The tasks at one sibling level combined by a synchronisation operator (|[]|) have to exchange information and are thus semantically closer to each other than the tasks combined by a choice ([]) or interleaving (|||) operator where the performances of the tasks are more independent.

4.2 Gathering information from temporal relations among tasks for structuring presentations

Temporal relationships among tasks can give useful information for structuring presentations other than the dialogues of the concrete user interface.
First of all, the temporal relation called enabling (>>) always indicates the border between two activation sets. Thus within an activation set, there will never be

tasks composed by an *enabling* operator which indicates that when a task terminates it activates another task.
Different presentations can be used to present two activation sets where the transition among them is determined by an enabling operator (Figure 4 shows an example where the user first specifies a query and next receives the related result):

- Both activation sets can be shown in the same presentation unit, this is suggested when we have sequential tasks with information exchange ([]>>) because this can mean that they are tightly related, especially when such sequential tasks have to be performed multiple times.
- The first activation set is presented in one presentation unit, and the second activation set is shown in a separate unit. The first presentation unit is no longer visible when the second presentation unit is shown. This is suggested when there are sequential tasks that are strongly unrelated to each other or the tasks require a high amount of information to be presented.
- An intermediate solution is that both activation sets are presented in separate presentation units, as in point 2, but here the first presentation unit is still visible but not reactive, while the second presentation unit is shown in a modal state. When the task associated with the second presentation unit is being performed the first presentation unit cannot be manipulated. However, if it is possible to perform multiple iterations of the two tasks then when the second task is terminated it is possible to activate again the first presentation unit.

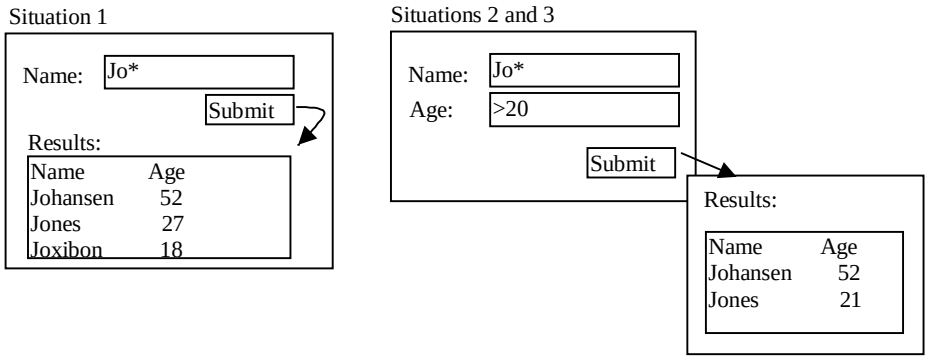

Figure 4: The three possible displays of two activation sets divided by an enabling operator

The disabling ([>) operator does not generate a division between the presentation of the tasks of the two activation sets, but they suggest how to structure the presentations of the tasks belonging to one activation set. They indicate a kind of group division within an activation set.
The tasks before the disabling operator can generally be considered as one group and thus be presented closer to each other. The control tasks, which perform the disabling, can be located so that they highlight their function of controlling the tasks which can be disabled and belong to the same activation set.
In the example of task specification in Figure 5 we can find one activation set {Enter Name, Enter Department, Submit} whose tasks are composed by one disabling operator and one interleaving operator. The Submit task applies to both the Enter Name and the Enter Department task.

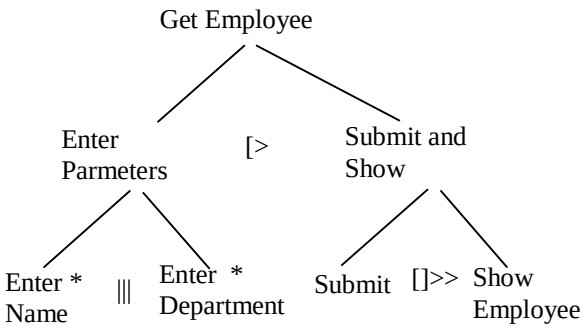

Figure 5: A task model containing a disabling and an interleaving operator

Figure 6 shows an example of a corresponding presentation structure. The advantage of this rule is that some control tasks can be automatically identified and then placed in a part of the presentation following the designer's choice.

Figure 6: Division into an activation set by the disabling operator

The interleaving operator (|||) does not provide any particular indication, whereas the use of the synchronisation (|[]|) operator shows that we have tasks that communicate with each other and thus need to be presented close together. This often happens when there are multiple tasks that allow the user to modify the same data.
When the choice operator ([]) occurs at the lowest level of the task tree it indicates choices that are strictly connected as they are grouped by some parent tasks, whereas if it occurs at the highest levels it indicates activities which are not too semantically related.

4.3 Relationships among Activation sets

Once we have identified activation sets and their main presentation structure, we have to take into account the problem of the transition between the presentations of two activation sets.
If we find that there is an intersection among the tasks belonging to the two activation sets then we can have a rule indicating that the presentation of the tasks included in both activation sets should be the same in order to keep consistency across different presentations. For example, the usual tasks allowing users to close or cancel an application should be placed in the same position with the same interaction technique. For the tasks which are no longer available, in the next activation set the related presentation resources can be reallocated for the new tasks introduced in the next activation set.

5 TASK TYPES AND DATA TYPES FOR PRESENTATION SELECTION

Once we have defined the structure of the presentation we have to define the type of presentation to associate with each basic task in the structure. In this case our approach is to create an environment where for each task, depending on its type, there is a set of predefined Presentation Templates which are suitable for supporting its semantics.
We now consider the basic tasks and we try to indicate their possible presentation, depending on their semantics and the type of data involved. Here we only consider single leaf tasks, and thus the related presentations are not particularly structured.
For the choice of the most suitable presentation some additional information might be needed about data cardinality.

5.1 Task Types

In the ConcurTaskTrees notation, four categories of tasks are recognised: *Abstract*, *User*, *Interaction* and *Application* task types. The *Abstract* task is used to indicate that it has subtasks of different types. The *Abstract task* is not directly applicable as an indicator for what type of presentation should be chosen for a specific task. This is because the abstract task is never a basic task.
The *User task* is a task type that does not require direct interactions with the user interface because the actions of this task type are cognitive actions which do not require the manipulation of any device such as identifying a strategy to solve a problem or looking at the screen or speaking with a colleague.
The two remaining task categories directly involve interactions with the presentation of the user interface. The *Interaction task* requires the user's initiative in interacting with the presentation objects.
The *Application task* is used to indicate activities completely performed by the application. The application task is often a system reaction to a user's action. There are also situations in the interaction where the application itself initiates actions, for example by giving the user an instruction or a suggestion for some action.

In the activation sets at the lowest level of the task tree only interaction and application basic tasks are included. The task classification needs to be further subdivided in order to give clearer indications of the semantics of the task, for this purpose we introduce a set of task types for each task category. Other specific information is also needed for the design of the presentation related to the task tree: the data types manipulated to perform the tasks.

The data type is an indicator of what type of presentation to choose. In Interaction tasks we want to consider the type of data that is the input to the application. For example, an "enter name" interaction basic task will be connected with the data type *string.* In Application tasks we want to consider the type of data which have to be presented.
For this selection of presentations, rules can be defined to (semi-)automatically choose the most suitable presentation for that specific task.

5.2 Interaction Tasks

In interaction tasks, the user takes the initiative in the interaction: s/he gives input to the application. In combination with the data types and cardinality, some basic presentation rules can be determined.
We can classify interaction tasks into Selection, Edit and Control task types.
A *Selection* task is very common in many applications. The user can select one or more items from a set or range of items. We can further classify this type of task depending on whether single or multiple selections are supported, and whether the selectable items are of the same type.
The presentation rules for selection take the data cardinality into consideration. In this case the data cardinality indicates the amount of data the user can select from. For example, if we consider a range of Integer value(s) then if it is a single selection with a small amount of data then a spin button is preferable, if a large amount of data is considered, then a data slider would be better. For multiple selections from a small amount of data, check boxes are preferred, while for multiple selections from a large amount of data, a listbox allowing the selection of multiple items is the best presentation.
Edit tasks are tasks that allow users to specify input data and this information can be modified before being definitively sent to the application. The presentation for editing is very simple to determine, since it depends on the data types associated with the information which has to be given as input to the application.
In *Control* tasks the user triggers actions explicitly. This means that this type of interaction task needs to be presented very clearly in the user interface, since triggers have important effects on the total task of the user. There is no data type involved in the trigger task, since the purpose of the control task is to generate a control event indicating when something should happen.
Several presentations can be used to trigger an action. Buttons, toolbar buttons, icons, hyperlinks and menu items can all be used to perform control tasks, directly needed by the user. Actions can also be activated by voice and gesture-based techniques.
The presentation of a control technique should attract the user's attention. Colour, size and font determine the appeal of the presentation. Another option is the use of colour to indicate that the user has already performed a certain trigger action, as with hyperlinks.

5.3 Application Tasks

Application tasks are used to indicate that the application performs an activity. In combination with the purpose of the presentation and the data types, some presentation rules can be defined.
There are various types of application tasks:

- *Overview*: the application shows a summary of a set of data, for example giving the minimum, average, and maximum values of the data considered.
- *Comparison*: the purpose of the presentation is to facilitate the user in comparing the values of some quantities of the same type, for example the revenues of different years.
- *Locate*: the application gives detailed information on a set of data so as to allow the user to rapidly find the desired information, for example emails received by name of sender, date or topic.
- *Grouping*: there is a one-to-many relationship among two data attributes which have to be presented at the same time and this relation has to be highlighted in the presentation (Aloia et al., 1998). For example, if the application has to present clients and sales orders the presentation should group the orders by clients.
- *Calculation*: the application performs some internal processing and gives feedback on the partial results; for example, when the application is searching for data that satisfy some criteria and it dynamically indicates the number of data found.
- *Application control*: a control event has to be generated after a predefined time and the application dynamically gives feedback on the time left before generating the control. For example, in computer-based training there may be an indication of the time available for completing an exercise.

6 PRESENTATION PATTERNS

Once the task types of the leaf tasks of the activation sets have been determined and the task model structure from which the activation set is generated is known, we can identify patterns of presentations. We can identify structures of presentations which can be considered task-oriented presentation patterns. In some cases combinations of these patterns can be used in a task model. The patterns that we have identified until now are:

- *Form*: contains combinations of Edit and Select tasks following indications from the structure of the task model such as possible groupings of tasks.

- *Control*: concerns the tasks that control the main current activities. These tasks can be triggered either by the user or the application, and they usually are in mutual choice.
- *Multiple Views*: tasks communicate with each other. There are some data which can be modified, either by the user or the application, and two or more representations of such data are updated and given to the end user.
- *Process&Present*: occurs when there are some sequential tasks that are semantically strictly connected. In this situation there is an application with some processing and results presentation. The user may want to perform these tasks several times, so s/he prefers to have them presented continuously even if they belong to different activation sets.

7 AN EXAMPLE

This section gives an example of a task model, and uses the proposed approach to determine the final presentation. The example chosen is taken from an application for Enterprise Resource Planning. We consider the possibility to search for some employees from a data base, and then select and edit the data of a particular employee.

In the task model we can first identify an iterative handling task which can be disabled by a close task. Then we can search for employees by entering parameters such as name and department. Once these parameters have been submitted, the application can calculate them and show the list of employees satisfying the criteria given. The user can select one of them. Both searching and selecting can occur many times without constraints until the user decides to edit either one selected item of information about an employee or some new information. During editing various items of information can be given until the user saves or cancels the modification.

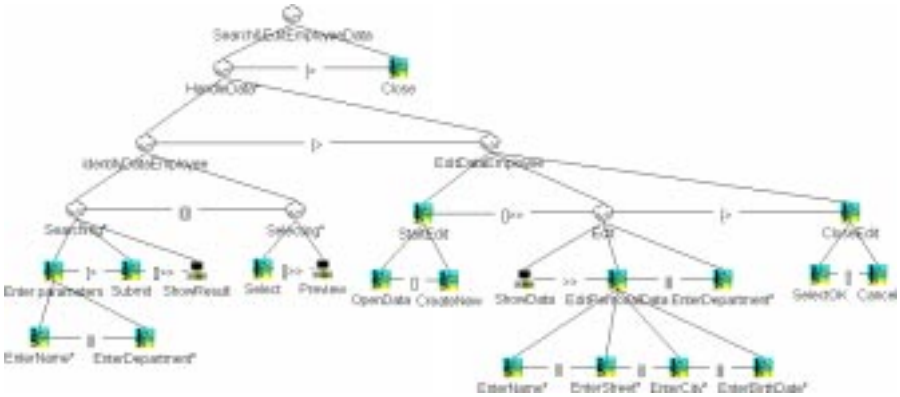

Figure 7: Search and Edit Employee Task Model Example

Once the activation sets of the example are identified by applying our algorithm we can start to consider how to design the presentation.

Figure 8: The activation sets of the example

The structure of the task tree highlights that there is an initial main grouping of tasks at the second level (those related to Search and Select employee and those related to Edit employee). In the first group of tasks we can further decompose it into two activation sets. Since these activation sets are tightly semantically related we are in the case of Process&Present, and we present them all together in the same presentation structure. In this structure there are couples of tasks which have to be presented in close proximity because they often communicate with each other, such as Select and Preview.

Figure 9: The two presentation structures obtained

In the second main structure there are two patterns: one Form which involves Editing tasks which can be logically grouped, and one Control which allows the user to determine whether or not to save the modifications.

8 CONCLUSIONS

In this paper we have shown the first results of an approach which aims to give systematic support for deriving information from task models which are useful for the design of presentations.
The proposed approach starts with a top-down analysis of the task model to define the activation sets of basic tasks. These activation sets give an indication of what basic tasks should be enabled during the same period of time and thus be supported by the same presentation. The second part of the approach is to determine the combined presentation of the basic tasks. We consider possible grouping of basic tasks identified using information in the task model itself. This grouping can be derived from the temporal relations involved in the task tree of the activation set and from the structure of the task model. The temporal relations can be useful to determine the design of possible presentations. Finally, how the basic tasks are presented depends also on their type.
We plan to further refine our approach in order to identify an extended set of rules which can be incorporated into an automatic tool which we have started to implement with the initial rules. This tool will be integrated with our current tool for task modelling (http://giove.cnuce.cnr.it/ctte.html). The purpose of the new tool is to exploit the knowledge of the information in the task model which for designing presentations. Such information will be given to designers in order to support their work while allowing them to tailor it for their specific application.
Further work will be dedicated to extend the taxonomy especially to take more into account the possibility of multimedia support of task performance.

9 REFERENCES

Aloia, N. Bendini, T. Paternò, F. Santoro, C. (1998) Design of Multimedia Semantic Presentation Templates: Options, Problems and Criteria of Use, Proceedings ACM AVI'98

Breedvelt, I. Paternò, F. Severijns, C. (1997) Reusable Structures in Task Models, Proceedings Design, Specification, Verification of Interactive Systems '97, Granada, June 97, Springer Verlag, pp.251-265.

Casner, S. (1991) A Task-Analytic Approach to the Automated Design of Graphic Presentations, ACM Trans. on Graphics, Vol. 10, N. 2, April 1991.

Castells, P. Szekely, P. Salcher, E. (1997) Declarative Models of Presentation, Proceedings ACM IUI'97, pp.137-144

Diaper, D. (1989) Task Analysis for Human Computer Interaction, Ellis Horwood, Books in Information Technology, 1989.

Johnson, P., Wilson, S. Markopoulos, P. and Pycock, J. (1993) "ADEPT, Advanced Design Environment for Prototyping with Task Models", Proceedings of INTERCHI'93, ACM Press, 56-57.

Mackinlay, J. (1986) Automating the Design of Graphical Presentations of Relational Information, ACM Transactions on Graphics, Vol.5, N.2, April 1986, pp.110-141.

Marcus, A. (1992) Graphic Design for Electronic Documents and user Interfaces, ACM Press, 1992.

Paternò, F. (1997) Understanding Task Model and User Interface Architecture Relationships, CNUCE Internal Report, December 1997.

Paterno', F. Mancini, C. Meniconi, S. (1997) ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, Proceedings Interact'97, Chapman&Hall, July'97. pp.362-369.

Roth, S. Kolojejchick, J. Mattis, J. Goldstein, J. (1994) Interactive Graphic Design Using Automatic Presentation Knowledge. Proceedings of ACM CHI'94 Conference, Boston, USA, April, 1994, pp.112-117.

Vanderdonckt, J. (1993) A Corpus of Selection Rules for Choosing Interaction Objects, Technical report 93/3, August 1993.

Vanderdonckt, J. Gillo, X. (1994) Visual Techniques for Traditional and Multimedia Layouts, Proceedings AVI'94, pp.95-104

Zhou, M. Feiner, S. (1997) Top-Down Hierarchical Planning of Coherent Visual Discourse, Proceedings IUI'97, pp.129-136