Applying Information Visualization Techniques to Visual Representations of Task Models

Fabio Paternò
ISTI-CNR
Via G.Moruzzi 1, Pisa, Italy
fabio.paterno@isti.cnr.t

Enrico Zini
University of Bologna
Bologna, Italy
zinie@cs.unibo.it

ABSTRACT

This paper shows how information visualization techniques can be used to improve the effectiveness of task model representations. In particular, we discuss how fisheye and semantic-zoom representations have been used to improve the effectiveness of the ConcurTaskTrees notation. The approach can also be useful for improving other visual modelling languages. We also report on a first evaluation of the proposed representations.

Author Keywords
Task model representations, information visualization, fisheye, semantic zoom, ConcurTaskTrees.
ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI).

INTRODUCTION

Nowadays, visual modelling is widely adopted in a variety of contexts. The set of notations provided by UML [8] represents the most evident case, but visual representations of many types of models are widely used. However, we are still far from visual representations that are easy to develop, analyse and modify, especially when large case studies are considered. As soon as the visual model increases in complexity, designers have to interact with many graphical symbols connected in various ways and have difficulties in analysing the specification and understanding the relations among the various parts.
In the human-computer interaction area one of the most recognised modelling activities is task modelling. Task models play an important role because they represent the logical activities that should support users in reaching their goals. Thus, knowing the tasks necessary to goal attainment is fundamental to the design process.
The need for modelling is most acutely felt when the design aims to support system implementation as well. If we gave developers only informal representations (such as scenarios or paper mock-ups), they would have to make many design decisions on their own, likely without the necessary background, to obtain a complete interactive system.

Task models describe the set of tasks supported by an interactive system and their relationships. Numerous task model formalisms and methodologies have been developed. However, one of the main problems in task modelling was that it is a time-consuming, sometimes discouraging process. To overcome such a limitation, interest has been increasing in the use of tool support. Engineered tool support in order to ease the development and analysis of task models and make them acceptable to a large number of designers has started to appear [7].

One issue is how to represent such models. Many proposals have been put forward to represent task models. Hierarchical task analysis [11] has a long history and is still sometimes used. The concept of hierarchical decomposition of activities to describe has shown to be successful because it allows designers to consider the various possible abstraction levels while still maintaining a clear indication of the relationships among them. However, just representing graphically the hierarchical structure may not be enough to obtain representations easy to analyse, especially when the specification becomes large.
The goal of this work is to investigate how information visualization techniques (such as semantic zooming and fisheyes [3]) can be tailored and applied to improve the effectiveness of the task model representations and associated environments, providing different interactive representations depending on the abstraction level of interest, or the

aspects that designers want to analyse or the type of issues that they want to uncover.

In the paper we first discuss related work, and we recall the main characteristics of the ConcurTaskTrees (CTT) notation and the associated environment for the readers who are not familiar with them. Then, we discuss the method followed to redesign the notation and the associated tool and present the new representation and the possible ways to interact with it. The final part is dedicated to reporting on a first evaluation of the new environment and providing some concluding remarks and indications for future work.

RELATED WORK

Many proposals have been put forward to represent task models. Hierarchical task analysis has a long history and is still sometimes used. More generally, such notations can vary according to various dimensions:

- *syntax (textual vs graphical)*, there are notations that are mainly textual, such as UAN [6] where there is a textual composition of tasks enriched with tables associated with the basic tasks. GOMS [1] is mainly textual, even if CPM-GOMS has a more graphical structure because it has been enhanced with PERT-charts that highlight the parallel activities. ConcurTaskTrees [9] and GTA [12], are mainly graphical representations aimed at better highlighting the hierarchical structure.
- *set of operators for task composition*, this is a point where there are substantial differences among the proposed notations. UAN and CTT are those that provide the richest set of temporal relationships. This allows designers to describe more flexible ways to perform tasks.
- *level of formality*, in some cases notations have been proposed without paying sufficient attention to define the meaning of the operators. The effect is that sometimes when task models are created, it is unclear what is actually being described. This is because the meaning of many instances of such composition operators is unclear.

If we consider visual representations of task models, it is possible to note that they often share the idea of providing a hierarchical representation. The main differences are in how such hierarchy is represented, how task names and associated operators can be composed and represented. For example, in ConcurTaskTrees the hierarchical structure is represented from top to down whereas in GTA it is from left to right. None of the proposals attempts to address issues related to when the specifications become large and the overall model does not fit well in a window or a screen. ConcurTaskTrees also associates an icon to each task to indicate its performance allocation. In Opta [13] the internal nodes of the hierarchical structure are associated with temporal operators instead of tasks.

THE STARTING POINT: CONCURTASKTREES AND THE CTTE TOOL

ConcurTaskTrees is a notation that focuses on activities. It allows designers to concentrate on the activities that users aim to perform, that are the most relevant aspects when designing interactive applications that encompass both user and system-related aspects. This approach allows designers to avoid low-level implementation details that at the design stage would only obscure the decisions to take.

It has a hierarchical structure (see an example in Figure 1) because a hierarchical structure is something very intuitive, in fact often when people have to solve a problem they tend to decompose it into smaller problems still maintaining the relationships among the various parts of the solution. The hierarchical structure of this specification has two advantages: it provides a wide range of granularity allowing large and small task structures to be reused, it enables reusable task structures to be defined at both low and high semantic level.

A rich set of possible temporal relationships between the tasks can be defined. This set provides more possibilities than those offered by concurrent notations, such as LOTOS. This sort of aspect is usually implicit, expressed informally in the output of task analysis. Making the analyst use these operators is a substantial change to normal practice. The reason for this innovation is that after an informal task analysis we want designers to express clearly the logical temporal relationships. This is because such ordering should be taken into account in the user interface implementation to allow the user to perform at any time the tasks that should be enabled from a semantic point of view.

How the performance of the task is allocated is indicated by the related category and it is explicitly represented by using icons.

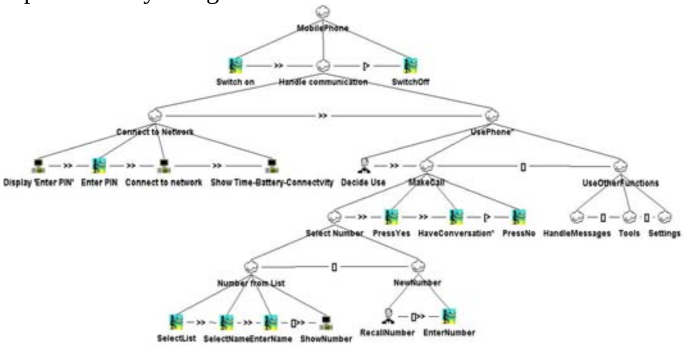


Figure 1: An example of task model in ConcurTaskTrees.

Once the tasks are identified it is important to indicate the objects that have to be manipulated to support their performance. Two broad types of objects can be considered: the user interface objects and the application domain objects. Multiple user interface objects can be associated to a domain objects (for example, temperature can be represented by a bar-chart of a textual value).

For each single task it is possible to directly specify a number of attributes and related information. There is one section on general information. It includes the identifier and extended name of the task, its category and type, frequency of use, some informal annotation that the designer may want to store, indication of possible preconditions and whether it is an iterative, optional or connection task. While the category of a task indicates the allocation of its performance, the type of a task allows designers to group tasks depending on their semantics. Each category has its own types of tasks. In the interaction category examples of task types are: selection (the task allows the user to select some information); control (the task allows the user to trigger a control event that can activate a functionality); editing (the task allows the user to enter a value); monitoring; responding to alerts. This classification is useful to drive the choice of the most suitable interaction or presentation techniques to support the task performance. Frequency of use is another useful type of information because the interaction techniques associated with more frequent tasks need to be better highlighted to obtain an efficient user interface. The platform attribute (desktop, PDA, cellular, …) allows the designer to indicate for what type of devices the task is suitable. This information is particularly useful in the design of nomadic applications (applications that can be accessed through multiple types of platforms). For each task, it is possible to indicate the objects (name and class) that have to be manipulated to perform it. Since the performance of the same task in different platforms can require the manipulation of different sets of objects, it is possible to indicate for each platform what objects should be considered. In multi-user applications different users may have different access rights.

The notation is supported by a tool, the ConcurTaskTrees Environment [7], which supports editing, analysis, and interactive simulations of the dynamic performance of sequence of tasks. As you can see in Figure 2 a small overview window is also provided on the top-right part to help in the analysis when large models are created and a general zooming facility is provided but there is no other technique applied able to support the adaptation of the presentation to the current focus of interest taking into account the structure of the visual specification. For this reason a new study has been carried out to identify innovative solutions for presenting visual models.

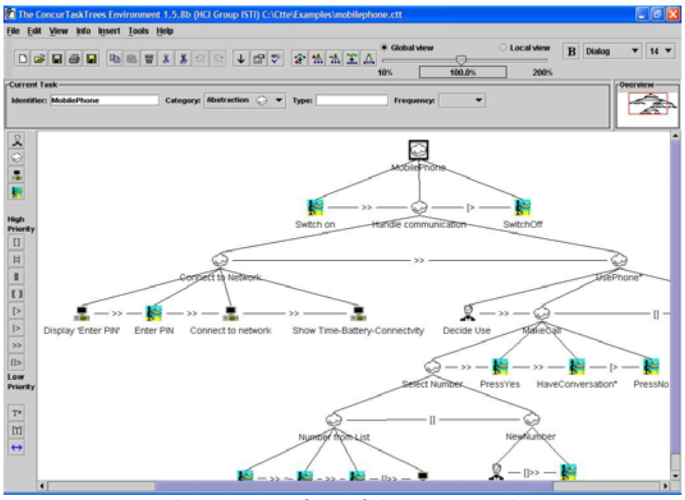


Figure 2: The user interface of the CTTE tool.

IDENTIFICATION OF REQUIREMENTS FOR THE NEW REPRESENTATION

The improvement proposal for the representation has been set up using a wide number of investigation methods, to take into account the many variables involved, being users, the notation structure itself and the tasks users perform with the notation. Since ConcurTaskTrees is a notation that has been adopted for a long time already, the investigation phase was also a good opportunity to collect the various pieces of feedback gathered in the past.

The first step in the investigation was to identify the user base working with ConcurTaskTrees and the CTTE environment. This was accomplished partly by interviewing people working in the research group that developed them and partly by analysing a web form people need to fill in before downloading the CTTE tool. The result was that users of CTTE are people doing user interface design, both in academic and in industrial environments.

The second step was to assess the goals of users in the identified target user base. To do this, a survey was compiled and sent via e-mail to many different user groups, and the results qualitatively analysed to pinpoint personal, job and practical goals. Beyond shedding light on users' goals, the survey has also provided a good source of information on performed tasks and frustrating aspects of the current interface.

Three major goals have been identified:

- Editing the model, a practical goal which is a needed pre-requisite for the other goals
- Analysing the model, a personal goal: once the model has been created, it is very useful to be able to use the tool to analyse data about it, explore it, understand it and making it "live" through the interactive simulator;
- Making a presentation of the model, a job goal: once the model has been studied, it is important to present the result of the study, including a representation of the model itself and the results of the investigation.

The resulting user and goal profile has then been used as a frame of reference to filter past feedback. All available sources have been scrutinized and relevant items have been extracted and collected, notably a user-centered evaluation, a Cognitive Dimensions [5] evaluation [2] and an Isometrics-based general evaluation of the CTTE tool and its included ConcurTaskTrees notation. The result of this work was a valuable collection of feedback, effectively sorted into areas of relevance to user needs. The last step in preparing the design of the new representation has then been to run two experiments to investigate how the current representation is perceived and the importance of its various parts.

The first experiment has been a "navigation test" we expressly designed for this purpose [14], which has been very effective in identifying the perceived importance of various aspects of the structure of the notation. The experiment involved using a photo-editing program to cover a ConcurTaskTrees model with an opaque layer, and then asking the subjects to use the "rubber eraser" tool in the opaque layer to uncover the model as they explored it (see Figure 3). The order in which elements were uncovered, as well as the elements that have not been uncovered, gave an interesting outline on the importance for the users of the various items and relationships among them. This method has proven to be an effective, cheap alternative to using sophisticated eye-tracking devices.

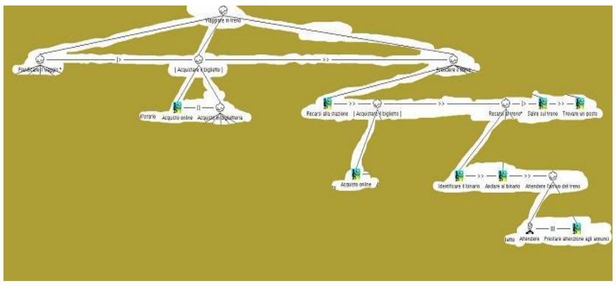


Figure 3: Example of final output of interactive uncovering.

The "navigation test" has shown that there is an important relationship between a task and the complete list of tasks in which it is decomposed. This relation is not always clearly provided in the original representation, and a possible improvement can be always keeping sibling tasks grouped and strongly connected with their parent task. Another finding is that the arcs joining a task with its children are not all needed, and could be removed to reduce visual clutter.

The second experiment has been a traditional "card sorting" experiment that identified a ranking among the perceived importance of the various pieces of information that can be provided in a ConcurTaskTrees model.

This rich body of data collected was finally enough to drive the design and evaluation of the new representation.

THE NEW PROPOSED REPRESENTATION

The resulting interface consists of different editing environments targeting the main user goals identified. The main environment is the Modelling Tool, for editing and analysing the structure. Two other editors are also provided: the Details Editor, for editing data and the Layout Editor, for modifying the layout to produce a pleasing presentation of the structure.

The Modelling Tools

The main result of the redesign has been the Model Editor, which introduces a novel approach to editing a hierarchical representation: the hierarchy can be edited with an agile interaction metaphor, featuring a text caret that can be moved around the representation and used to insert, cut, paste and rename tasks and task hierarchies.

The Model Editor also uses a completely automatic tree layout that allows editing the model structure without needing to manually control the placement of the various items on the screen.

One of the thorniest problems of the old notation was that big models consisting of many levels of task decompositions would require a great amount of space to be laid out, well beyond the capacity of a computer screen, making working with large models cumbersome and even largest ones totally unfeasible.

The redesign has addressed this issue by introducing three changes in the representations:

- Task labels are now word-wrapped, to reduce horizontal requirement of space, which most impacted tree growth;
- All the tasks involved in a task decomposition are grouped together, as suggested by the "navigation test". This avoids dispersing the elements in which a task is decomposed, as the task expression

as a whole has been found to be the main mean of understanding the decomposed task;

- The introduction of a multi-layer "fisheye view" within the task tree ultimately allows editing arbitrarily large trees, by working on a portion of the model while still keeping the perception of the whole tree structure.

Other simple techniques have been followed to further improve the tree comprehensibility:

- the set of arcs emanating from a task to all its decomposing tasks, deemed unnecessary in the "navigation test", has been replaced by a single arc, which divides into a set of arcs only at the very end to keep the decomposition metaphor. This reduces the amount of items on screen, both removing unneeded visual noise and reducing cognitive load.
- greyed parenthesis are used to explicitly mark when the operator precedence makes the order of evaluation of temporal operators not flowing naturally from left to right. This eases the hard cognitive problem of remembering operator precedence with the eight different temporal operators to be able to assert the behaviour of a task.

Icons have been redesigned for clearness using a set of requirements we designed:

- symbols must be immediately understandable, as one of the goals of the notation is to be communicative;
- symbols must be such that they are easily recognised in black and white, so that they can be reproduced via monochrome printing or drawing using pencil and paper;
- symbols must be quick and easy to draw by hand;
- symbols should use existing graphic signs, to allow existing social conventions to be reused in their decoding;
- symbols must use the least possible number of different graphical signs, to avoid high cognitive load in their decoding;
- symbols must be similar, but different: similar in order to recognize them as part of a homogeneous group, and different to avoid confusing them with each other.

While all these improvements are a key part in making the notation more effective, the fisheye view is the ultimate step in making large trees comfortably workable.
The original idea of fisheye [3] is based on defining three functions over the viewable elements of the notation:

- "Level of Detail" (LOD(x)) defines the degree of generality/specificity of the element, increasing as the element describes a more specific part of the whole
- "Distance from focus" (d(x)) defines the distance from an element x to the element which is the current centre of focus
- "Degree of Interest" (DOI(x)), computed as DOI(x)=h(f(LOD(x))+g(d(x))), where f, g and h are monotonically increasing functions; it defines a measure of how "interesting" the current element is, considering the focus of the user's attention to be centred in a specific place.

Once the DOI function is defined, the original fisheye view algorithm works by simply showing those elements whose DOI values are greater than a given threshold value t.

In a task tree, the function d(x) can be defined as the "walking distance", or shortest path, between the centre of the focus and the node x. The current fisheye implementation in CTTE has shown to give good results with a simple definition of DOI as DOI(x)=-d(x). Furthermore, the fisheye algorithm has been improved by introducing an intermediate level between full show and pruning, and by representing the pruned tree structure (using a semantic zooming technique) instead of totally hiding all the pruned trees.

The resulting algorithm is as follows:

- the centre of the focus is represented by the node at the caret;
- nodes with a high DOI value (DOI(x)>t) are represented in full size;
- nodes with a DOI value equal to the threshold value (DOI(x)=t), are represented in a smaller size;
- all the other nodes are replaced by a small outline of their tree structure (no node is left out).

Figure 4 shows how the task model represented in the traditional CTT representation in Figure 1 can be displayed using the new extension that supports fisheye representations. In this case the focus is on the New Number task. It is possible to see that the part of the hierarchical structure that involves such task is highlighted, whereas the more remote parts are represented in small sizes. In some cases the names are not reported at all. In the bottom-right part there is also a small overview window that indicates through red dots the tasks that are currently highlighted in the fisheye representation. In the bottom left part, some details regarding the current task focus are given as well.

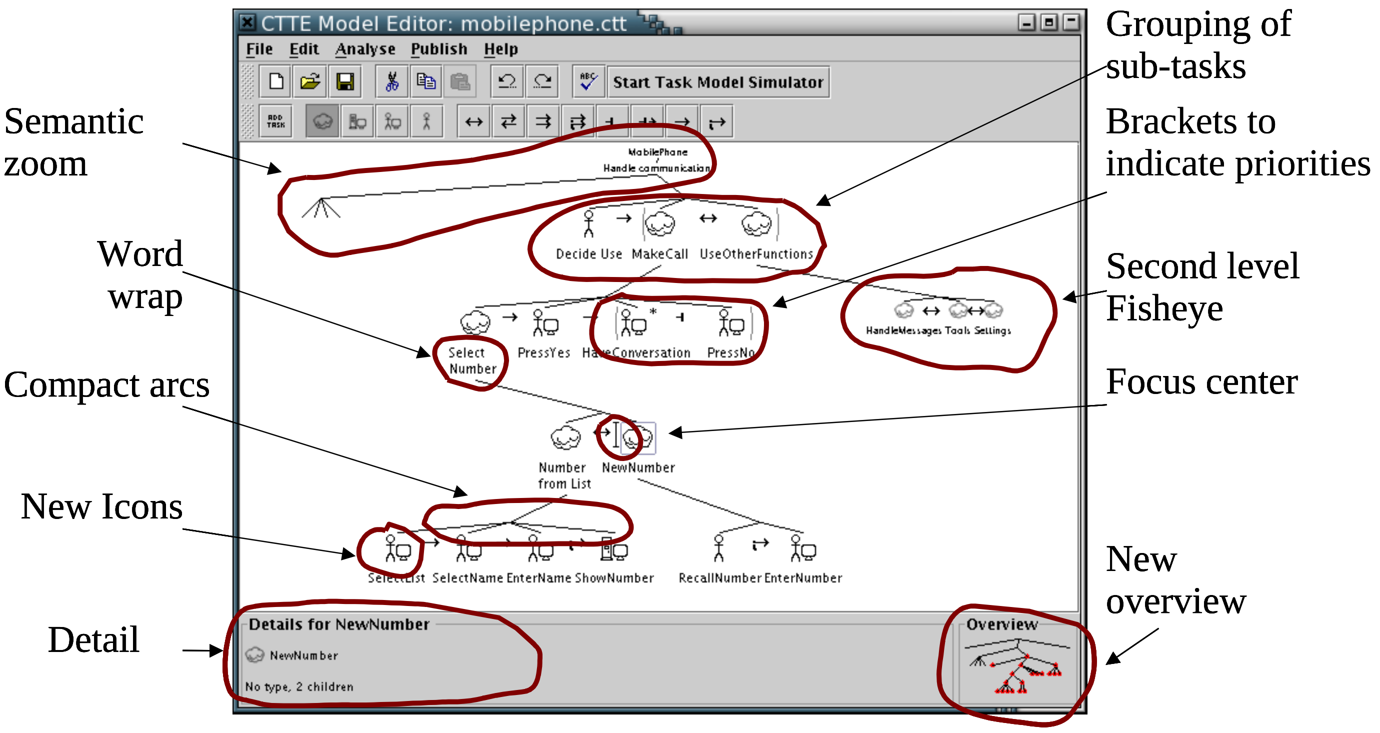


Figure 4: An example of fisheye applied to the task model representation.

While the cursor is moving around the tree, the fisheye is continuously recomputed to move the focus centre on the new position, and both the tree structure in the overview and the tree structure of the pruned branches in the representation can be clicked to bring nodes to the centre of focus. This gives the impression of a completely perceivable and easy-to-manipulate model independent of its size and current focus.

This approach has also been applied to the rendering provided by the CTTE simulator. This is a useful feature to analyse the dynamic behaviour represented by the task model. The tool allows the designer to select a task and then shows the enabled tasks after its performance according to the temporal relations defined in the model. This also allows designers to interactively identify potential scenarios supported by the current task model. In the new interface for the simulator two windows are presented: one small one showing the overall model and the list of enabled tasks (the user can select one of them to carry on the interactive simulation) and a large one applying the fisheye representation with multiple foci (Figure 5).

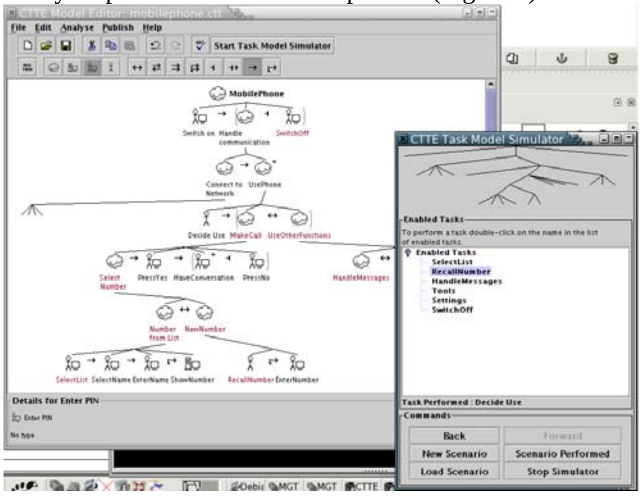


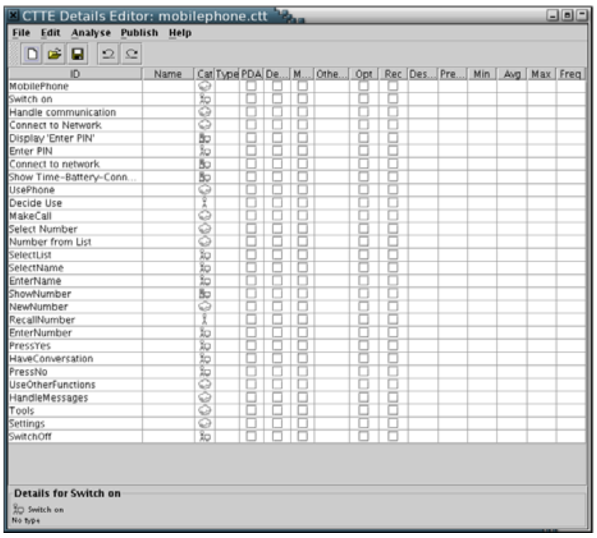
Figure 5: The fisheye approach applied to simulator output.

The fisheye implementation provided in the task model simulator is different and benefits from a distinct strategy: when the simulator is operative, the fisheye algorithm considers every expression containing enabled tasks as a focus center, and all the other expressions are hidden and replaced with a sketch of the subtrees that contain them. This allows designers to keep a clear idea of the simulator state, reducing the encumbrance of the parts of the task tree that are not currently relevant.

The Details Editor

The second metaphor provided for model editing is the Details Editor, which allows designers to work with the task details without being encumbered with the model structure. The Details Editor (see Figure 6) offers the metaphor of a spreadsheet, with one task per row, which allows designers to have a full view over the entire task data set.

This view provides "visibility" [5] to a big amount of data that was previously hard to access except on a task-by-task basis, and opens new possibilities of exploration of the task data as a whole: sorting the columns, for example, can be used to highlight relationships and trends among the data, as one would normally do with familiar table views.



CTTE Details Editor: mobilephone.ctt

File Edit Analyse Publish Help

ID	Name	Cat	Type	PDA	De...	M...	Othe...	Opt	Rec	Des...	Pre...	Min	Avg	Max	Freq
MobilePhone				☐	☐	☐		☐	☐						
Switch on				☐	☐	☐		☐	☐						
Handle communication				☐	☐	☐		☐	☐						
Connect to Network				☐	☐	☐		☐	☐						
Display 'Enter PIN'				☐	☐	☐		☐	☐						
Enter PIN				☐	☐	☐		☐	☐						
Connect to network				☐	☐	☐		☐	☐						
Show Time-Battery-Conn...				☐	☐	☐		☐	☐						
UsePhone				☐	☐	☐		☐	☐						
Decide Use				☐	☐	☐		☐	☐						
MakeCall				☐	☐	☐		☐	☐						
Select Number				☐	☐	☐		☐	☐						
Number from List				☐	☐	☐		☐	☐						
SelectList				☐	☐	☐		☐	☐						
SelectName				☐	☐	☐		☐	☐						
EnterName				☐	☐	☐		☐	☐						
ShowNumber				☐	☐	☐		☐	☐						
NewNumber				☐	☐	☐		☐	☐						
RecallNumber				☐	☐	☐		☐	☐						
EnterNumber				☐	☐	☐		☐	☐						
PressYes				☐	☐	☐		☐	☐						
HaveConversation				☐	☐	☐		☐	☐						
PressNo				☐	☐	☐		☐	☐						
UseOtherFunctions				☐	☐	☐		☐	☐						
HandleMessages				☐	☐	☐		☐	☐						
Tools				☐	☐	☐		☐	☐						
Settings				☐	☐	☐		☐	☐						
SwitchOff				☐	☐	☐		☐	☐						

Details for Switch on

Switch on

No type

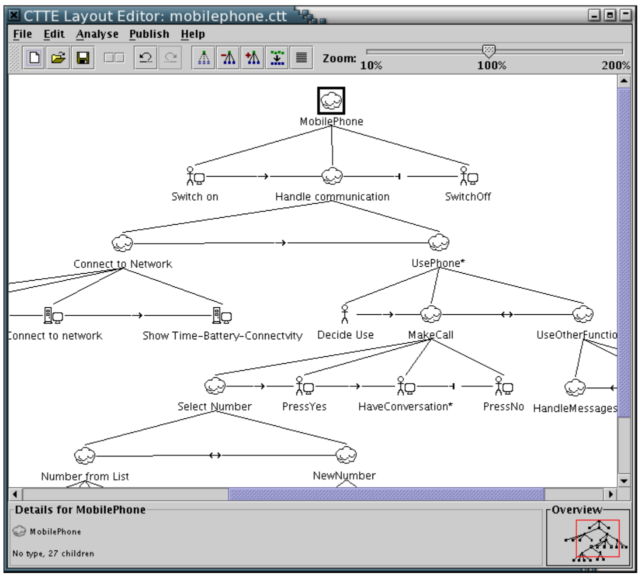
Figure 6: The details editor.

The Layout Editor

The third editing metaphor is the Layout Editor (see Figure 7), which allows designers to control the final tree layout and design before publishing, using the interaction pattern of a vector graphics tool.

While the Layout Editor is not a required element for modelling and analysis, presenting the results has been identified as an important user goal during the initial investigation phase. This understanding allowed us to decide to gather former CTTE manual layout functionalities instead of removing them, and organize them into a separate, specific environment, ready to be evolved and integrated to form a powerful and focused accessory to the modelling tool.

Although more research and development is needed on the Layout Editor, it seems a good candidate as the starting point for adding communication aids such as colour highlighting, arrows and annotations to task models.



Details for MobilePhone

MobilePhone

No type, 27 children

Figure 7: The layout editor.

EVALUATION OF THE REPRESENTATION

After the implementation of the designed improvements reached the level of a working prototype, we went on to a first evaluation of the effectiveness of the modifications made. The evaluation involved eight young computer science researchers whose characteristics conformed quite well to the prototype user we had identified initially.

The evaluation consisted of a 20-minute modelling exercise, followed by a feedback survey. The modelling task requested involved a familiar domain to allow the subjects to focus on interacting with the tool instead of solving complex modelling issues. The only training provided was a one-page sketch introduction to the new modelling interface. During the exercise, a moderator (the second author) was available to answer volunteers' questions, taking notes of the issues raised.

The evaluation gave positive, encouraging results:

- Most of the times the facilitator has been called during the exercise was to report implementation bugs, but there have been no major issues with understanding the new interface;
- 6 over 8 volunteers continued playing with the new interface even after the allocated 20 minutes expired, showing how the prototype gathered considerable interest;
- Besides some reserve and reports of difficulties and possible improvements, the first impression given on the survey was unanimously positive;
- When asked what were the most interesting and what were the most useful features, most of the answers indicated the new representation even if other major and long-requested improvements such as multi-document editing and full unlimited undo also appeared in the new CTTE;
- Even though no question has been made in the survey about the efficiency of editing, when asked what old problems are solved by the new interface, 5 over 8 volunteers reported a clearer, faster, more functional or more immediate editing;
- When asked about frustrating aspects of the new interface, most of them regarded implementation bugs or yet unimplemented editing features, which were naturally present due to the tool being in a prototype phase. It is very encouraging that frustrating aspects have mainly regarded the impossibility of fully use the new interface instead of problems potentially introduced by the new way of interaction;
- 6 over 8 volunteers declared they would keep using the new interface;
- 6 over 8 volunteers declared they would suggest the new interface to experienced CTTE users;
- 7 over 8 volunteers declared they would suggest the new interface to new CTTE users.

This first evaluation exercise mainly aimed to understand whether the new environment can satisfy designers. A more extended testing is required to better evaluate various usability aspects.

CONCLUSIONS

We have presented an original solution for applying information visualization techniques to representing and manipulating task models and thereby improve their effectiveness. This solution has been applied to the ConcurTaskTrees notation for task models. A tool supporting the editing and analysis of models in this new representation has been developed as well. The results of a first evaluation carried out with a number of user interface designers and developers have been positive.

To our knowledge, no other solution using similar techniques has ever been applied to any task model representation. The techniques proposed may also be interesting for other visual modelling notations, such as those used in the UML approach.

Future work will be dedicated to more extensive usability testing and investigating the application and tailoring of other information visualization techniques to task models.

REFERENCES

1. Card, S., Moran, T., Newell, A., The Psychology of Human-Computer Interaction, Lawrence Erlbaum, Hillsdale, 1983.
2. Fairburn, C., Goillau, P., Wright P., CTT Notation Evaluation, MEFISTO Working Paper N.4-9.
3. Furnas, G. (1981). The FISHEYE view: A new look at structured files, technical Memorandum #81-11221-9, Bell Laboratories, Murray Hill, New Jersey 07974, USA, 12 October 1981.
4. Gamboa Rodriguez F., Scapin D., Editing MAD* Task Description for Specifying User Interfaces at both Semantic and Presentation Levels, Proceedings DSV-IS'97, pp.193-208, Springer Verlag
5. Green T.R.G., Petre M.; Usability analysis of visual programming environments: a 'cognitive dimensions' framework, in J. Visual Languages and Computing, Vol.7, N.2, pp.131-174, 1996.
6. Hartson R., Gray P., “Temporal Aspects of Tasks in the User Action Notation”, Human Computer Interaction, Vol.7, pp.1-45, 1992.
7. Mori, G., Paternò, F., Santoro, C. (2002). CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Transactions on Software Engineering, pp. 797-813, August 2002 (Vol. 28, No. 8).
8. OMG Unified Modeling Language Specification; available at http://www.omg.org/technology/ documents/formal/uml.htm
9. Paternò, F., Model-Based Design and Evaluation of Interactive Application. Springer Verlag, ISBN 1-85233-155-0, 1999.
10. Paternò, F. (2003), ConcurTaskTrees: an engineering notation for task models. In Dan Diaper & Neville A. Stanton (Eds.) The Handbook of Task Analysis for Human Computer Interaction. London: Lawrence Erlbaum Associates,483-503.
11. Sheperd (1989), Analysis and Training in Information Technology Tasks, in D.Diaper (ed.), Task Analysis for Human-Computer Interaction, chapter 1, pp.15-55, Ellis Horwood, Chichester, 1989.
12. van Welie M., van der Veer G.C., Eliëns A., “An Ontology for Task World Models”, Proceedings DSV-IS'98, pp.57-70, Springer Verlag, 1998.
13. Violante F., “OPTA: Open Task Analysis”, Ph.D. Thesis, Politecnico di Milano, Dipartimento di Elettronica e Informazione, 2001
14. Zini E. (2004), Ricerca e prototipazione di rappresentazioni efficacy per la modellizzazione visuale: il caso di ConcurTaskTrees Environment. Master Thesis. University of Bologna, 2004.