Describing Just-UI Concepts Using a Task Notation

Pedro J. Molina[1], Jorge Belenguer[1 2], and Óscar Pastor[2]

[1] CARE Technologies S.A.
Research & Development Dpt.
Pda. Madrigueres, 44.
03700 Denia, Alicante, Spain.
{pjmolina|jbelenguer@care-t.com}
http://www.care-t.com
[2] Technical University of Valencia
Dpt. of Information Systems & Computation
Camino de Vera, s/n,
46002 Valencia, Spain.
{jorbefa|opastor}@dsic.upv.es

Abstract. A proposal for describing Just-UI patterns using a task oriented notation (ConcurTaskTrees) is presented. This work provides a method to convert a given pattern into a task tree and the contrary, to detect Just-UI patterns in task models. This bridge makes it possible to establish links between concepts from one model to the other one.

1 Introduction

Progressively, patterns are becoming good building blocks for developing User Interfaces. One of the key properties of a pattern is that it contains distilled well-proven concepts from experience senior developers that can be reused. Pattern collections like Welie, Tidwell, Javahery or, Molina [18, 15, 5, 8] are good examples of this usage.

However, patterns tend to be described and documented in a very informal way, using natural language, following the Alexander format [1].

This informal description is not a problem if the intended use of the patterns is only for teaching novice developers or HCI students. Nevertheless, computer supported patterns used for code generation like Just-UI [8, 9] needs a more precise description to be interpreted by software tools like validators or code generators.

Therefore, the objective of this paper is to explore the possibility of describing conceptual user interface patterns using a formal based task model notation. Providing in this way, a formal framework to describe precisely the patterns task semantic.

Just-UI [8, 9] is a proposal for building abstract user interface specifications. It is composed of a collection of patterns organized to conform a pattern lan-

guage useful for describing abstract user interfaces in the domain of business information systems.

Due to their empiric nature, patterns arise from the sequence of the following processes: observation, experimentation, and finally abstraction. In this sense, it is difficult to formalize such patterns. However, we think that using formal methods to describe patterns could help to unambiguously explain the semantic captured by such patterns.

In this way, Task Analysis techniques are helpful to describe the tasks implicated in the user interface. Several approaches for task models exist: from low-level tasks models like GOMS [2] to high-level like GTA [16], MAD [14] or CTT [12]. Moreover, a standardization effort has been proposed in [6]. The *ConcurTaskTree* (CTT) [12] is a complete task notation, based on a formal basis and supported with freely available tools [13]. Such a notation is an excellent syntactic sugar that hides the complexity of the LOTOS formal language [4].

Therefore, we propose to use the ConcurTaskTree notation to describe the task semantics of the Just-UI concepts. We think that such a work will help to understand the semantics of the related concepts. Furthermore, each implementation of an abstract specification must be compliant with the model and respect such a semantic.

This paper is organized as follows: First of all, the Just-UI model and CTT notation are briefly introduced. Next section presents a mapping for each Just-UI pattern to CTT notation. Later on, a compositional algorithm is provided to build a full CTT tree for a complete Just-UI specification. Finally related work and conclusions are given.

2 The Just-UI Model

Just-UI [8, 9] is a Pattern-Based User Interface Model designed to specify user interfaces at the conceptual level (analysis level). The model is oriented to information system domain. It has been proven useful for producing fast prototypes [7] using code generation techniques. The model is composed by a pattern language structured in tree layers (access, interaction units and elemental patterns) as shown in Figure 1. For a detailed description, refer to [8].

3 The ConcurTaskTree Notation

ConcurTaskTree [12] is a well accepted notation in the UI field used for the specification of task models. The notation is supported by the semantics of the formal language LOTOS [4]. Moreover, Paternò and his colleagues have developed a CASE tool (ConcurTaskTree Environment [13]) to support the diagramming with this notation. In their own words [11]: *"this is the most engineered tool for task modelling and analysis and it is publicly available"* and *"it has been used in many countries in a number of university courses for teaching purposes and for research and development projects"*.

The notation defines four categories of tasks:

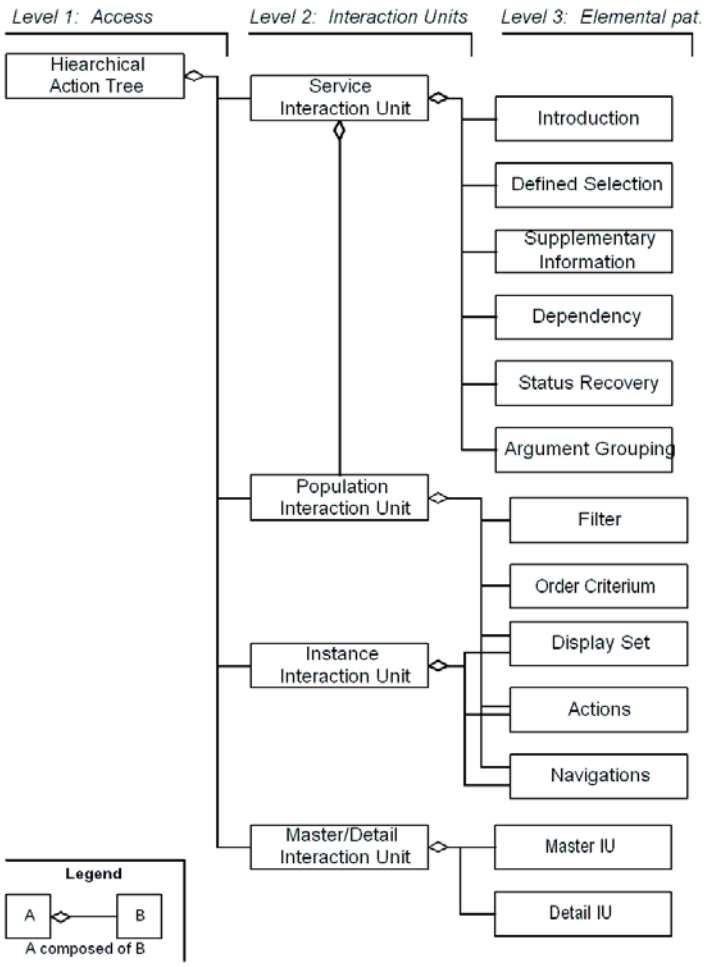

Fig. 1. Just-UI pattern language overview.

1. **User Tasks:** Tasks performed by the user. Usually, they are cognitive activities like thinking a strategy to solve a problem.
2. **Application Tasks:** Tasks performed by the application. For example: presenting results to the user.
3. **Interaction Tasks:** Tasks performed by the user interacting with the system. For example: pressing a button or a key.
4. **Abstract Tasks:** Tasks which requires complex activities or that could be decomposed in simpler ones.

ConcurTaskTree uses a graphical representation for each category of task as shows Figure 2.

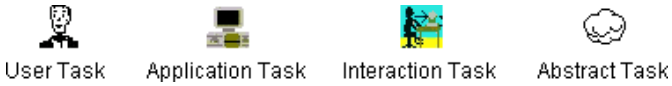

Fig. 2. Categories of tasks in ConcurTaskTrees.

A task can be decomposed in subtasks. Such subtasks are related among them by means of temporal relationships. The ConcurTaskTrees notation provides a set of temporal operators based in the semantics of LOTOS. The operators are described in Table 1.

4 Just-UI to CTT Mappings

In this section, JUST-UI concepts are going to be introduced and expressed using the CTT notation, thus allowing us to make use of the semantics that underlies the LOTOS formal language, and of the free-distribution author tool available for this notation (ConcurTaskTree [13]).

The main idea behind Just-UI is to use small pieces to build the specification, reusing such pieces to build complex components.

Following this idea, we propose to use small CTT sub-trees to describe the task semantic of each small pattern in Just-UI and afterwards, to build for a given Just-IU specification a unique CTT tree describing the whole task semantic of the specification.

In order to success in this goal, we will define in each CTT subtree *Connection Points*.

Definition 1. ***Connection Point***
A node of a CTT subtree is a Connection Point iff:

1. *The node is a Interaction Task leaf node labelled with a UML stereotype-like syntax* ``<<xxxx Link>>''.
2. *No other node can be considered to be a Connection Point.*

Interleaving $(T_1 \vert\vert\vert T_2)$	Tasks may occur in any order without constrains.
Order Independence $(T_1 \vert = \vert T_2)$	Subtasks may occur in any order but not at the same time.
Choice $(T_1 [] T_2)$	Choice from a set of task. Once a task is selected the other tasks in the set are not available until the selected task is completed.
Concurrency with information exchange $(T_1 \vert [] \vert T_2)$	Two tasks may execute concurrently but they have to synchronize in order to exchange information.
Deactivation $(T_1 [> T_2)$	The task T_1 is definitively deactivated once task T_2 starts.
Enabling $(T_1 >> T_2)$	Task T_1 enables the occurrence of T_2 once T_1 has finished.
Enabling with information passing $(T_1 [] >> T_2)$	Task T_1 provides more information to T_2 that just enabling it.
Suspend-Resume $(T_1 \vert > T_2)$	Task T_2 interrupts task T_1. When task T_2 ends, task T_1 can resume its execution in the same point it was interrupted.
Iteration $T*$	The task is executing continually. When the task ends, the task is executed again. This cycled ends when the task is interrupted by other task.
Finite interaction $T(n)$	It is used when it is known in advance how many times a task will be performed.
Optional task $[T]$	Task may be executed or not.
Recursion	A subtree originated by the task contains another occurrence of it. This can be useful to specify a recursive task until a another task interrupts the recursion.

Table 1. Temporal operators in ConcurTaskTrees.

4.1 Service Interaction Unit

The Service Interaction Unit allows the user to provide the arguments needed to launch a service in a scenario. The interaction tasks that can be performed by the user are:

- to input the value of a parameter,
- to launch the requested service, and
- to cancel the invocation of the service.

Figure 3 shows the corresponding CTT tree for Service Interaction Unit. The user has to introduce the values for the parameters of the service. The user can select any parameter without restrictions and then change its value. At any moment, the user can trigger two actions: `Close` (cancels the execution of the service closing this interaction unit) or `Launch Service` (checks the parameters

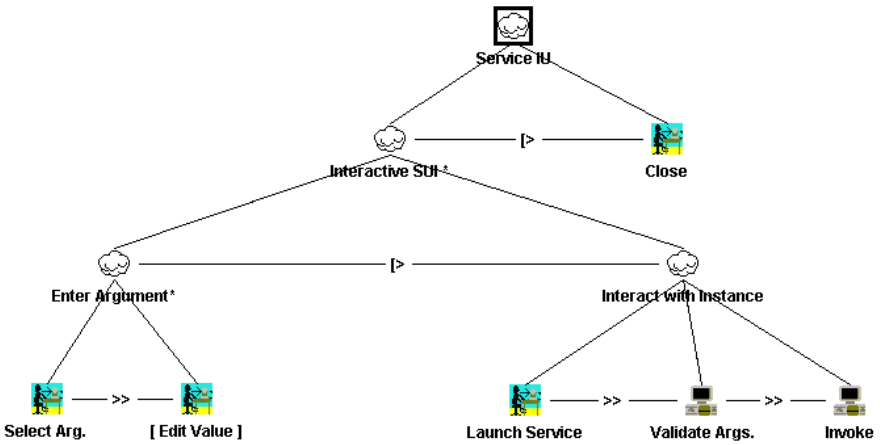

Fig. 3. CTT task tree for Service Interaction Unit.

and launch a request for service execution). In this case, the systems performs the validation task and finally, invokes the requested functionality.

Due to other parts of the application will provide access to Service Interaction Units, the only Connection Point available in this CTT subtree is the root node.

4.2 Instance Interaction Unit

The second type of interaction unit is strongly oriented to object observation tasks. The user tasks involved are:

- selecting an object to work with,
- observation,
- requesting a change in the object's state,
- demand related information, and
- abandon the interaction unit.

The corresponding CTT tree (Figure 4) shows how tasks are temporally related. The user can observe the object's data, request a change, demand more information or close the interaction unit (without restrictions). At any given time, the object to show can be selected.

The root node is a Connection Point (other parts of the UI will access to this interaction unit). The leaf nodes `<<Action Link>>` and `<<Navigation Link>>` are also Connection Points to Actions and Navigation pattern, respectively.

4.3 Population Interaction Unit

The Population Interaction Unit deals with collections of objects. The user tasks involved are:

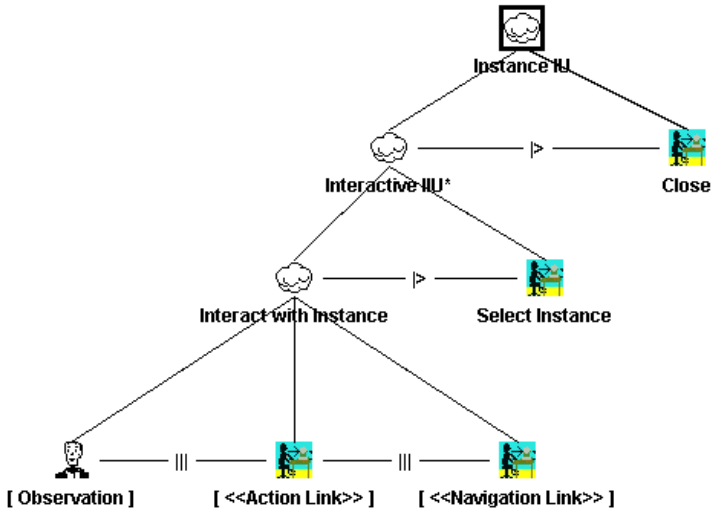

Fig. 4. CTT task tree for Instance Interaction Unit.

- selecting an order criterium,
- selecting filter criterium,
- observation,
- selecting an object to work with,
- requesting a change in the object's state,
- demand related information, and
- abandon the interaction unit.

The CTT tree (Figure 5) shows the tasks involved in the scenario and its temporal relations. A user can search for objects, select an object and finally interact with it. The searching task involves selecting an order criteria, a filter, fill in the filter variables and finally, search. Once the results appears, the user can select an instance and interact with it.

Similar to the Instance IU, the root node is a Connection Point (other parts of the IU will access to this interaction unit). The leaf nodes `<<Action Link>>` and `<<Navigation Link>>` are also Connection Points to Actions and Navigation pattern, respectively.

4.4 Master/Detail Interaction Unit

The Master/Detail Interaction Unit is composed by several interactions units. The details units are full synchronized with the master unit.

The corresponding CTT tree (Figure 6) is extremely simple. The details tasks are synchronized with the master task using the temporal operator [] >> indicating *enabling with information exchange*. At any time, the user could interrupt the dialogue closing the interaction unit.

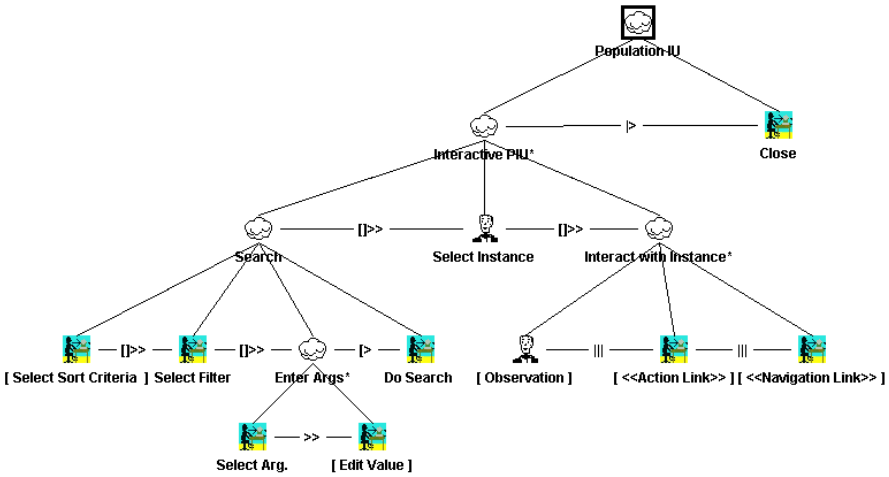

Fig. 5. CTT task tree for Population Interaction Unit.

The Connection Points in this CTT tree are:

- The root node.Other parts of the UI will access to this interaction unit.
- The leaf node `<<Master UI Link>>`. Connects with an Interaction Unit playing the master role.
- The leaf node `<<Detail UI Link>>`. Connects with an Interaction Unit playing the detail role.

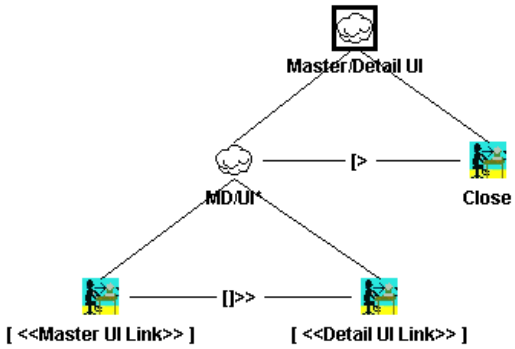

Fig. 6. CTT task tree for Master/Detail Interaction Unit.

4.5 Actions

Actions are a set of elements that jump to new interaction units to launch services and change the object's state. Actions are pure choices ([]): the user can select an action that will direct him to another interaction unit. The CTT tree is very simple (Figure 7): contains a leaf node for each offered action using the temporal operator choice ([]).

The Connection Points here are: the root node (instance and population interaction units use them) and each of the selectable action items (navigable jumps to target interaction units).

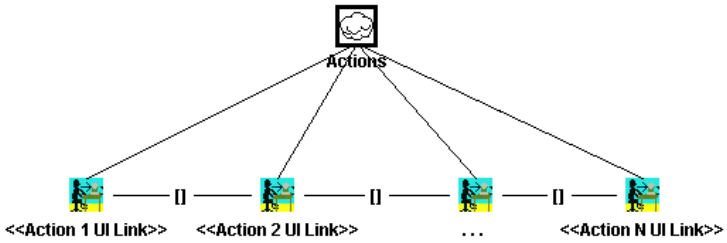

Fig. 7. CTT task tree for Actions.

4.6 Navigation

Navigation presents to the user a set of links to reach information related to the current object. Again, navigation is a pure choice. The user can select a navigational item that will direct him to another interaction unit to show the related information. The CTT tree is similar to the previous one (Figure 8): it contains a leaf node for each navigational item using the temporal operator [] (choice).

The Connection Points here are similar to Action ones: the root node (instance and population interaction units use them) and each of the selectable navigation items (navigable jumps to target interaction units).

4.7 Hierarchical Action Tree

The Hierarchical Action Tree[3] models the access to the application. It distributes the functionality of the application using the *Gradual Approach Principle*[3]. The HAT is a tree where intermediate nodes are labels to organize the functionality and leaf nodes contains the functionality with links to the corresponding interaction units.

[3] Henceforward, referred as HAT.

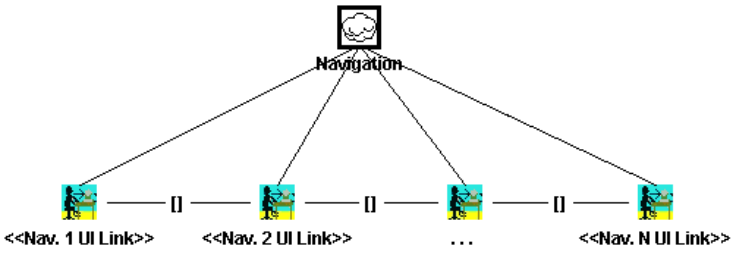

Fig. 8. CTT task tree for Navigation.

In this way, the associated CTT tree can be expressed as a choice in the tree(see Figure 9). The user will select a leaf node using a menu like widget. Although we could model the intermediate nodes, this was not done for simplicity (only leaf nodes were considered). The result tree is again a pure choice tree where lead nodes represents the leaf nodes in the HAT tree

The root node is a Connection Point. Moreover, its the root of the User Interface specification CTT tree. Each HAT element are also Connection Points to other Interaction Units.

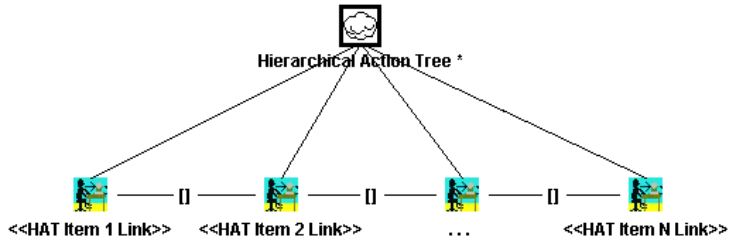

Fig. 9. CTT task tree for Hierarchical Action Tree.

5 UI Specification as Composition of Subtrees

The complete semantics of the JUST-UI specification can be defined as the composition of the semantics given for each concept described previously as follows:

Proposition 1. *The complete tasks specification is a CTT tree where:*

1. *The root tree is a CTT tree corresponding to a HAT tree for the considered view.*

2. *For each* HAT *items Connection Point of the previous CTT tree, substitute it using the corresponding CTT subtree for target interaction units.*
3. *For each* Action *Connection Point of the previous CTT tree, substitute it using the corresponding CTT subtree for Actions.*
4. *For each* Navigation *Connection Point of the previous CTT tree, substitute it using the corresponding CTT subtree for Navigation.*
5. *For each* Nav_i *and* Action_j *items Connection Points of the previous CTT tree, substitute it using the corresponding CTT subtree for target interaction units.*
6. *For each* Master *and* Detail *Connection Points of the previous CTT tree, substitute it using the corresponding CTT subtree for target interaction units.*
7. *Apply recursively steps numbered 3, 4, 5 & 6 until there is no more connection points as leaf nodes.*

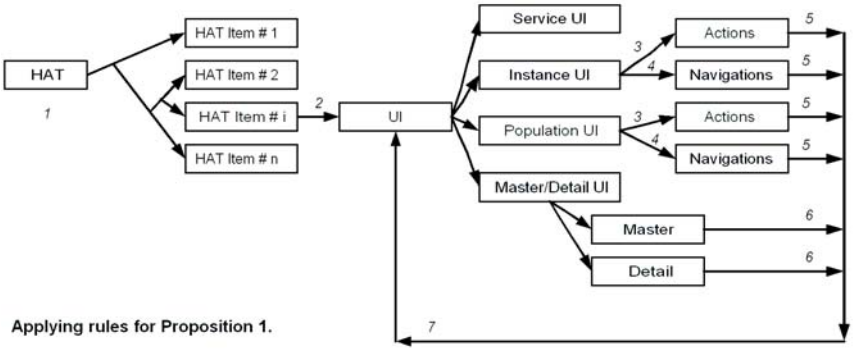

Fig. 10. Rule order evaluation.

Figure 10 shows a graphical representation of the rule evaluation. The CTT tree built in this way, expresses the task semantics of the user interface specified in the Just-UI model. This CTT tree can be directly translated to its correspondent LOTOS specification. And therefore, it can be validated, animated or analyzed with the corresponding available tools for the LOTOS language.

5.1 Termination problem

The CTT formalism has a tree structure. However, Just-UI model is based on directed-graphs (where interaction units are nodes and where navigation and actions are arcs). Obviously, the algebraic rules states that a cycled graph cannot be represent using a finite tree.

If the JUST-UI model has cycles, the tree obtained using the Proposition 1 its not finite.

However, its not necessary to build the whole tree. Cycles can be detected and the expansion in such point pruned. Whenever a user interact with a User Interface, he performs a finite walk-thought in the tree. Therefore, in animation and simulation tools, the expansion can be done incrementally when it is needed.

6 Related Work

Limbourg et al. [6] proposed a common meta-model for unifying different Task Models. Paternò [12, chapter 7] describes some patterns in interactive applications in terms of CTT. Later on, Nunes [10] provided examples of pattern hunting in CTT specifications for Wisdom concepts. This work shows the CTT equivalent tree form for Just-UI patterns. Moreover, Winckler [17] formally described some primitive tasks for web modeling using CTT, thus allowing to reuse them to describe more complex user tasks.

7 Conclusions

A method for describing the user task semantics embedded in the specification concepts used in Just-UI has been presented. The CTT notation has been used with success to reach this goal. The corresponding CTT sub-trees for each concept have been presented. Afterwards, an algorithm for building a CTT tree for a Just-UI specification using a compositional approach has also been presented.

This work is helpful for:

1. describing the task semantics of patterns in a more formal way that natural language,
2. expressing with a precise model the task semantics of a Just-UI specification,
3. providing a way of mapping Just-UI concepts to task trees,
4. validating Just-UI specifications using CTT based tools [13], and
5. automatically finding Just-UI patterns in CTT specifications (like pattern matching approaches).

In medium to big sized systems, task trees are big enough to be directly manipulated. Using a pattern approach like the one presented here could help to reduce the overload in the analysts' and designers' minds.

The presented work provides a formal framework to describe precisely the task semantic of the patterns. This is crucial in an approach like Just-UI, where patterns are used for code generation to multiple devices like Desktop, Web or PDA applications.

As future work, we will extend this approach to the whole patterns described in the Just-UI pattern language. However, the hard work is done with this job. The rest of patterns seem to be easier to describe in terms of a task notation.

Of course, others pattern collections could be mapped to CTT using this proposed strategy. This will improve the description of the patterns with a precise task specification.

8 Acknowledgments

Thank to Prof. Jean Vanderdonckt and Prof. Fabio Paternò for reviewing and providing useful comments to improve this work.

References

1. C. Alexander *"A Pattern Language: Towns, Buildings, Construction"* Oxford University Press, 1977.
2. S.K. Card, T.P. Moran, and A. Newell. *"The Psychology of Human-Computer Interaction"*. Lawrence Erlbaum Associates, Hillsdale, 1983.
3. IBM. *"Object Oriented Interface Design: IBM Common User Access Guidelines"*. Que, Carmel, Ind. USA, 1992.
4. ISO, *"Information Process Systems - Open Systems Interconection - LOTOS - A Formal Description Based on Temporal Ordering of Observational Behaviour"*, ISO/IS 8807. ISO Central Secretariat, 1988.
5. H. Javahery and A. Seffah *"A Model for Usability Pattern-Oriented Design"* TAMODIA'200. 1st International Workshop on TAsk, MOdels and DIAgrams for user interface design, Costin Pribeanu and Jean Vanderdonckt (editors), July, Academy of Economic Studies, pages 104-110, Bucarest, Rumania, July, 2002.
6. Q. Limbourg, C. Pribeanu, J. Vanderdonckt *"Towards Uniformed Task Models in a Model-Based Approach"* In Procs of: Interactive Systems: Design, Specification, and Verification. 8th International Workshop, DSV-IS 2001. C. Johnson (Ed.). LNCS 2220, p. 164 ff, Springer Verlag, 2001.
7. P.J. Molina, S. Martí, and O. Pastor *"Prototipado rápido de interfaces de usuario"* (In Spanish) (Rapid Prototyping of User Interfaces) In Proceedings of IDEAS 2002. ISBN: 959-7160-14-5. La Habana, Cuba. April, 2002.
8. P.J. Molina, S. Meliá, and O. Pastor. *"JUST-UI: A User Interface Specification Model"*. In Proceedings of CADUI'2002 Conference, pages 323-334, Valenciennes, Francia, May, Kluwer Academics, 2002. Extended version to appear in Journal of Human-Computer Interaction (RIHM).
9. P.J. Molina, S. Meliá, and O. Pastor *"User Interface Conceptual Patterns"*. In Proceedings of DSV-IS'2002 Conference, pages 201-214, Rostock, Germany, June, 2002. Also in LNCS 2545, Springer Verlag, December 2002.
10. N.J. Nunes. *"Object Modeling for User-Centered Development and User-Interface Design."* PhD Thesis. University of Madeira, Portugal, 2001. Available at `http://math.uma.pt/njn/NunoPhDPrint.pdf`.
11. F. Paternò. *ConcurTaskTrees and UML: how to marry them?* In Proceedings of TUPIS '00 at UML'00, Tupis 2000.
12. F. Paternò. *Model-Based Design and Evaluation of Interactive Applications.* Springer, 2000.
13. F. Paternò. *"CTTE. The ConcurTaskTree Environment"*, Available on-line at `http://giove.cnuce.cnr.it/ctte.html`, 2001.
14. D. Scapin and C.Pierret-Golbreich. *"Towards a Method for Task Description: MAD"* In: Berlinguet, L., Berthelette, D. (eds.): Proc. of Conf. Work with Display Units WWU'89, Elsevier Science Publishers, Amsterdam 27-34, 1989.
15. J. Tidwell *"Common Ground: A Pattern Language for Human-Computer Interface Design"* Available at: http://www.mit.edu/ jtidwell/common_ground.html 1999.

16. G.C. Van der Veer, B.F. Van der Lenting, and B.A.J. Bergevoet. *"GTA: Groupware Task Analysis - Modeling Complexity"* Acta Psychologica 91, 297-322, 1996.
17. M. Winckler, P. Palanque, C. Farenc, and M. Pimenta. *"Task-Based Assessment of Web Navigation Design"* In Proceedings of TAMODIA '02 (Bucharest, Romania, July 2002), 161-169. ISBN: 973-8360-01-3, 2002.
18. M. van Welie *"The Amsterdam Collection of Patterns in User Interface Design"* Available at: http://www.cs.vu.nl/ martijn/patterns/index.html 2000.