

# The ConcurTaskTrees Boundary Object

Fabio Paternò

ISTI - C.N.R.

56124 - Pisa, Italy

fabio.paterno@isti.cnr.it

## ABSTRACT

In this paper I discuss a possible example of boundary objects: the ConcurTaskTrees notation. It is a notation for specifying task models and has tool support. I have been working in several projects involving people with different background, different domains, and different goals. Thus, I can briefly report on related experiences in using such notation as a kind of boundary object.

## Keywords

Task models, notations, tools, multi-disciplinary projects.

## INTRODUCTION

I started to work on the ConcurTaskTrees notation in about 1995. The goal was to obtain a notation to support designers of interactive systems. It also had to be able to specify flexible behaviour and possess a rigorous underlying definition of its elements. The result was a notation for task models able to graphically represent the hierarchical structure, with a set of temporal operators able to describe concurrent behaviour (including interruptions), icons for indicating task performance allocation, several attributes associated with the main task properties, and indications of the objects that should be manipulated in order to accomplish the tasks. The notation also supports the possibility of describing task models of cooperative applications. This is achieved by developing the task model associated with each role involved and then describing the cooperation through a task model that specifies the constraints among tasks performed by different users. An extended description of the notation and related methods is in [1].

The notation has mainly been refined in a couple of European projects, during which a tool for supporting ConcurTaskTrees task model development, analysis, and interactive simulation has been developed. One project was MEFISTO (<http://giove.cnuce.cnr.it/mefisto.html>), a multi-disciplinary project aimed at developing methods and tools for design of safety-critical interactive applications. The project involved software developers, software designers, psychologists, and air traffic controllers. The notation was used by people with expertise in these fields, except the air traffic controllers. In general, involving air traffic controllers was difficult because their time is particularly expensive, and so we decided that in order to save time designers should have been a kind of intermediary between

such users and the notation and its application. Its use by software developers and designers was easier because they are used to more structured notations and can more easily learn them. An interesting application was the task modelling of one existing air traffic control application, because it raised a number of design issues that even people who have been working in the domain for a long time did not realise before.

The other project (GUITARE, <http://giove.cnuce.cnr.it/cameleon.html>) considered a different application domain, ERP (Enterprise Resource Planning) applications. It grew out of a previous collaboration between my group and a large (at that time) company in the ERP area (BAAN company). The collaboration was activated by a member of the company with background in the humanities. She liked the notation because she was able to understand it and it allowed her to specify in a logical manner the desired behaviour of the interactive applications to designers and developers. Even the software developers liked the approach because they saw the possibility of creating development environments able to start from logical descriptions and automatically generate consistent interfaces for their large applications.

During these projects we improved the environment supporting the notation because I felt it was strategically important to facilitate its use and adoption. The result is the ConcurTaskTrees Environment [2]. Initially the idea was to facilitate the development of models and support saving them in XML format. It was the first notation for task models described in XML format (this feature has been available since 1998). Then I realised that this was not enough. There is also a need for tool support in order to analyse the content of task models, to check whether they describe the desired behaviour and to think about how to use the information that they contain for supporting evaluation or design. An example is the interactive simulator that allows the designer to iteratively ask what the next enabled tasks are if a given task is performed. This provides an insight to the dynamic evolution of the activities described by the model that can be difficult to achieve by just looking at the model specification.

These two trial projects gave rise to a more refined notation [3] and a mature tool for external use. The tool was soon made publicly available (<http://giove.cnuce.cnr.it/ctte.html>), and it has been used in several countries and contexts. During the period 11 September 2002 -25

November 2003 there have been 1182 downloads, including by companies such as Nokia, Philips, Motorola, Samsung, IBM, BMW. It is interesting to see how different people have used it in different ways (even if a complete study on this aspect has not been carried out). The people who have considered ConcurTaskTrees range from formal methods experts to human factor experts. They often had very different goals. Some formal experts integrated it in their tools for performing rigorous demonstration of formal properties. Jacques Hugo, a Human Factors Engineer from South Africa working in the area of safety-critical systems, said in one email "All I can tell you is that in my experience task analysis is one of the most difficult and time-consuming tasks in human factors engineering. One of the main reasons for this difficulty is that it relies completely on the skill, experience, knowledge and time of the analyst. There is real need in the industry for a reliable tool to assist in the analysis and documentation of the most important phase in the process of application development."

One issue often raised was how to move from a conceptual analysis and specification to a corresponding real implementation. Often people like to see the result of their conceptual analysis quickly, otherwise they get frustrated. More generally, people like to move back and forth between conceptual and concrete representations when analysing design solutions. The possibility of addressing this issue was offered by the CAMELEON project (<http://giove.cnuce.cnr.it/cameleon.html>). This is another European project (October 2001-September 2004) aiming to develop tools and methods to support the design and development of context-dependent interfaces, in particular to adapt to different devices while still supporting usability criteria.

In this project we have designed and developed a tool (TERESA, <http://giove.cnuce.cnr.it/teresa.html>) [4] that is able to take a ConcurTaskTrees specification and support its transformation first into an abstract user interface and then into a concrete one and lastly into the user interface implementation taking into account the features of the device at hand. Currently, it supports this transformation into Web interfaces for desktop systems, mobile phones and vocal interfaces. In this way, the ConcurTaskTrees notation becomes an object immediately useful for software developers. They can first think in terms of logical activities. The notation also allows them to indicate for what platforms are suitable for the performance of each task. Then, the TERESA environment supports the generation of the corresponding interfaces. This generation process can be performed with different automation levels. It is possible to follow an almost completely automatic process where the tool takes the design decisions or a semi-automatic process where the tool suggests solutions but the designers/developers can modify them according to their needs. A study of the tool was carried out at Motorola Italy, who compared it with traditional approaches (such as a template for the design phase and Microsoft Front Page or Netscape Composer for the implementation phase). Results

showed similar total times for the traditional and TERESA approaches, with opposite impact on different phases. The use of the tool almost doubled required time at the design stage, while at the development stage the results showed dramatically improved prototyping performance, reducing the time required to half. This leaves a margin for further improvement, since the design time required by the TERESA approach is expected to decrease as the subjects become more familiar with model-based techniques and notations. Moreover the slight total time increase is acceptable since it involves a trade-off with overall design quality: all the subjects appreciated the benefits of a formal process and the support offered in identifying the most suitable interaction techniques. The evaluators noticed and appreciated the improved structure of the presentations and more consistent look of the pages resulting from the model-based approach.

A follow-up idea is to extend such approach in order to achieve natural development [5]. This means to ease the development process of interactive software systems through the use of familiar representations and intelligent environments able to map them onto corresponding implementations of interactive systems. This can enable end-users, people who are non-professional developers, at some point to create or modify a software artefact.

## CONCLUSIONS

In this paper, I have briefly discussed why a notation for task modelling and the related tool can be considered an example of boundary object: it has been used by people with different background either separately or in joint work. I have also briefly discussed some projects where this notation has been used with different perspectives and in different domains. Future work will be dedicated to extending the notation to obtain more effective representations.

## REFERENCES

1. F. Paternò, *Model-based Design and Evaluation of Interactive Applications*, Springer Verlag, 1999.
2. G. Mori, F. Paternò, C. Santoro, "CTTE: Support for Developing and Analysing Task Models for Interactive System Design", *IEEE Transactions on Software Engineering*, pp. 797-813, August 2002, IEEE Press.
3. F. Paternò, *ConcurTaskTrees: An Engineered Notation for Task Models*, Chapter 24, in Diaper, D., Stanton, N. (Eds.), *The Handbook of Task Analysis for Human-Computer Interaction*, Lawrence Erlbaum Associates, Mahwah, 2003
4. G. Mori, F. Paternò, C. Santoro, "Tool Support for Designing Nomadic Applications", *Proceedings ACM IUI'03*, Miami, pp.141-148, ACM Press.
5. F. Paternò, *From Model-based to Natural Development*, *Proceedings HCI International 2003*, Universal Access in HCI, pp.592-596, Lawrence Erlbaum Associates, Publisher.