# Improving Support for Visual Task Modelling

Fabio Paternò, Carmen Santoro and Lucio Davide Spano
CNR-ISTI, Via G. Moruzzi 1,
56124 Pisa, Italy
{fabio.paterno, carmen.santoro, lucio.davide.spano}@isti.cnr.it

**Abstract.** ConcurTaskTrees (CTT) and its supporting environment (CTTE) have been widely used for a significant period of time. However, users have expressed various concerns regarding their usability. In this paper, we present the modifications made so as to provide more effective support. In particular, the environment has been enhanced in order to make it more suitable for designing real-world applications, including improved support for task model editing and early prototype generation. We also report on two evaluation tests that provided useful feedback in order to decide how to improve the environment.

**Keywords:** Task models; Visual Tool Support; CTT

## 1 Introduction

Task models provide structured representations of how activities should be carried out in order to reach users' goals. They can be seen as a "lingua franca" between the various stakeholders in the development of interactive systems (users, designers and developers, to mention just a few). On the one hand, they are high-level descriptions that are comprehensible even to people without a programming background. On the other hand, they provide precise requirements for user interface software development as well. Over the years various notations for task modelling have been proposed together with the tools to support their development and analysis. However, often these approaches have been used mainly by the groups who developed them, and the associated tools were perceived as not mature enough. In this regard ConcurTaskTrees (CTT) [7] and the associated ConcurTaskTrees Environment (CTTE) [6] are an interesting exception. The environment made it possible to graphically represent, edit, export, interactively simulate, and check the consistency of task models represented in CTT.

The community using, extending, and applying CTT has steadily increased over the years, and various research contributions have involved CTT: some were aimed at extensions in order to increase its capabilities, while others applied it to new application domains. For the sake of brevity we mention only a small number of them. A more extended list is available at http://hiis.isti.cnr.it/tools/CTTE/CTT_publications/publications.html.

An example of a new application domain for task modelling is discussed in [1], where CTT is applied to serious games for training nurses. An example of a CTT extension is [4], which provides ideas to improve the scalability of the notation. The

COMM notation [2], instead, is aimed at extending the CTT notation in order to improve support for designing multi-user and multimodal systems. In [3], the consideration of contextual aspects in task modelling has led to the development of another extension to CTT.  In CTT the context is mainly considered from the multi-device point of view, with the possibility to specify which tasks are supported by specific platforms. Other task modelling notations have considered contextual aspects to some extent in their approaches (e.g. [5]). The notation is currently being considered for standardization, and an initial version of a standard for task models based on CTT has been published by W3C[1]. It overcomes limitations of previous standards [5].

In this paper, we report on the environment evolution, which was based on some usability evaluations, and discuss the changes that have been made, in particular to the associated visual tool.

## 2 Key Concepts of the CTT Notation

We first introduce a few key concepts of the CTT notation in order to better understand the concepts discussed in the paper. Further details on CTT can be found in [17]. CTT is a notation for describing task models. CTT tasks have a tree-based representation for their hierarchical structure: the children of a given task represent a more detailed description of their parent task. Tasks at the same level are connected through temporal operators. Each task belongs to one *category*: i)*user* for internal cognitive activity, ii)*application* for only-system performance, iii)*interaction* for tasks involving both user actions with associated system feedback, iv)*abstraction* for tasks that have sub-tasks belonging to different categories. The task *type* allows designers to classify tasks depending on their semantics. Each category has its own set of task types. A task can be associated with one or multiple domain objects that it manipulates. Each task can also have some properties: platform(s) (those suitable for its performance), informal description, and so forth. It is also possible to have a temporal operators among tasks, and even unary operators (iteration and optionality).

An important issue in designing task model notations is to find the right trade-off between expressiveness and complexity. One extension to improve the expressiveness of CTT has been the introduction of pre and post conditions, specified according to an appropriate syntax. Indeed, the execution of a task is often subject to the availability of a given resource or depends on the value of a certain variable. In order to express these dependencies, the CTT language already contained a precondition attribute in the task definition. However, the field was just a simple string, and the designer was not bound to any formalism in order to express the conditions, which were often specified in natural language. Despite its flexibility and the ability to describe these conditions in a human-readable way, its lack of formalism is not suitable for an automatic tool support. For instance, one of the most appreciated features of CTTE is the opportunity to simulate the execution of a task model so as to identify modelling errors or to show the support for a given scenario. Informal specifications of the preconditions would not allow coherent simulation because of possible ambiguities.

[1] http://www.w3.org/TR/task-models/

Although other ways of modelling conditions and constraints already exist (see e.g. OCL, a formal high-level language used to describe properties on UML models) we preferred to define a simple and flexible solution in order to find a trade-off between the capability of expressing conditions and the ease with which these conditions can be handled. In CTT a pre or post condition (which can be associated to a task) is a Boolean expression that is obtained by applying logical operators to constants and/or objects values. Their hierarchical structure allows the representation of complex Boolean conditions.

## 4 Evaluation

In this section we report on two evaluation tests that were conducted in different contexts in order to better assess the effectiveness of the tool. One involved undergraduate students and the other a group of designers recruited from an industrial project.

The students participating in the first evaluation had no particular UI development experience. However, before the test, they were given a quick introduction to the key concepts of UI modelling, and CTT/CTTE in particular. The goal of this test was to assess whether students were able to quickly become sufficiently proficient with the notation/tool to be able to model an interactive system of low-medium complexity. The test took place during a lecture and the users were observed by one of the authors.

The second evaluation involved developers/designers working in companies, who therefore had very different characteristics, time constraints and motivations compared to the students. They had, on average, more experience in UI development/programming but, differently from the students, they had not attended any specific training on UI models (apart from the basic knowledge gained in the project). Their motivation was understandably lower, as they were willing to spend only a short amount of time learning the features of the notation/tool. Moreover, the evaluation with the second group of users was conducted remotely so, it was not possible to observe the users.  Information was provided in a written format and communication was mainly made by email.

### 4.1 University Course Test

The goal of this test was to understand whether people without experience in task modelling could create CTT task models with minimal training/assistance.

**Participants.** 20 undergraduate students in Humanities Computing took part (11 females). They had low familiarity with models but they were trained before the test.

**Test Material and Procedure.** The students first attended a 90 minute lesson on requirements, scenarios, task analysis and modelling during which CTT and its tool support were introduced. Then, they were allowed 90 minutes to carry out the tasks required by the test. In the case where the exercise was not completed within the time, they could submit the built task model later via email. During the 90 minutes in the lab, one of the authors observed the class as they worked, and provided help, if requested, by explaining aspects related to the notation as well as the tool.

**Test Tasks.** Students had to develop the task model of an existing or future interactive computer-based system. The model was to have at least 3 levels of task decomposition, and include at least 15 tasks. In addition, it had to be correct according to the language (there is an automatic feature in the tool to check this property). The task type was to be indicated at least for the interaction and system tasks.

**Results.** All 20 students were able to complete the exercise and provide meaningful and correct task models. The correctness was assessed both syntactically, through the tool, and semantically (we manually checked the meaningfulness of the models). Despite having the same background, students' performance was very different, ranging from one who finished in 60 minutes to one who submitted it after a couple of weeks. Six students finished the exercise in the 90 minute session, and the others terminated it at home and sent the results by email.

The class exercise was also a good opportunity to observe some usability issues. One problem was adding the temporal relations into a CTT model. In the tested CTTE version users had to select the left sibling task and then select the operator to be included on the right hand of the selected task using a vertical bar. However, the observer noticed that some students tried to graphically select the two tasks by boxing them with the cursor and then attempted to insert the operator between them (by selecting it from the related bar). We realized that the students' behaviour was more intuitive then the selection process implemented at that time in the tool, so we decided to add this feature to the tool.

One aspect that disoriented some students was a tool feature which automatically changed the category of a parent task depending on the category of subtasks that were dynamically added. This feature had been introduced to ensure that the CTT specifications are correct according to the language. However, this proactive tool behaviour was perceived as slightly intrusive/unexpected by some users, who as a result might not have completely understood the rules for defining the task categories. Thus, we decided to disable this automatic support, and introduce it only on request by the user. The students also expressed the desire for some kind of rough preview of the resulting user interface. Accordingly, we introduced this feature, as explained in the following sections.

### 4.2 Industrial Project Test

The introduction of the preconditions was also stimulated during an industrial project, in which a multi-device (desktop and mobile) application for monitoring health was developed. Through this application users receive messages/reminders/notifications regarding their health, monitor how active they have been in the past, and how consistently they are taking medicines. Some preconditions were included in the task model and modelled by using the mechanism previously described. We performed a test to see whether developers/designers were able to understand/modify these model-based descriptions.

**Participants.** Volunteers were recruited through the project network (an email message was sent to the project mailing list). In the end, the test involved 8 people (3 female) from Italy, The Netherlands, Finland, and Belgium (average age: 35.7, with a range of 27-59). Six users held a degree, 2 were postgraduates. 5 users worked in research and 3 in an ICT company. User's knowledge of UI development was just a little above average (M= 3.13; SD= 0.83) on a scale from 1 to 5, where 1 is the most negative score (very bad), and 5 the most positive one (very good). Users' familiarity with models was very low (the majority had never created models before).

**Test Material and Procedure.** The test was conducted remotely: users performed it in their own environment and using their own equipment. Participants were emailed a document with some background information about task modelling and a task list to carry out. They were given a pre-built CTT task model, and instructions to download the CTTE tool. After completing the tasks, the testers filled in a questionnaire, divided into: i)a section collecting some user background information (gender, age, education, familiarity in developing applications through the use of models); ii)a section focusing on CTT/CTTE use, where users commented on the task model; iii) a section in which users provided further feedback. To fill in the questionnaire, the participants used the same 1-5 semantic scale as before. The modified material (the CTT task model and the filled questionnaire) was then returned to the evaluators.

**Test Tasks.** The tasks were selected so that users would explore and analyse both the language (CTT) and the corresponding tool (CTTE). The task model considered for the test was a simplified version of the model for a healthcare application (desktop platform), which also contained some preconditions already modelled in it. For the test, users had to import the task model within the CTTE tool and to analyse it. Then they performed the following two tasks: i) add the possibility for users to monitor glucose level to the model; ii) add the possibility that the system sends a notification to the user whenever the glucose level goes beyond a certain threshold.

**Results.** Regarding the effectiveness of the tool, all the users -apart from one- managed to provide syntactically correct task models. Just one user built a task model with one missing temporal relationship. We also assessed the semantic correctness of the task models produced. The following aspects were checked: i) whether the added tasks were correctly included within the provided task model (e.g. in the right place); ii) the appropriateness of the temporal relationships included; iii) whether the category for the included tasks was meaningful; iv) whether users provided a proper refinement of the higher level tasks that have to be included in the task model. On the one hand, users performed the highest number of errors while setting the task category (3 users wrongly used task category). On the other hand, users made the fewest errors while positioning the tasks within the task model (only 2 users made this mistake in one of the tasks) and while refining the specification of the two high-level tasks to be added in the task model (only 2 users did not provide a complete description of the required tasks). 3 users successfully completed all test tasks and provided an error-free task model. 4 users correctly completed the first task, 4 users did the same with the second task.

Participants also had to rate the comprehensibility of CTT (M = 3.13; SD= 1). Only one worst score was given (1= "very unclear"), commenting that he did not think of user interactions in terms of task trees. 3 participants found the underlying CTT logic quite clear and the tool easy to use (and they gave 4 marks to this aspect) thanks to the graphical representation used. This was true even in the case where participants had little expertise. However, one tester was confused by the way the CTTE tool handles the "interaction" and "user" task categories. The remaining 4 people rated this aspect with a 3 ("neutral" rating). Among them, one expressed annoyance with the fact that the automatic check changed the category of tasks to ensure the consistency of the specification (this was also mirrored in the highest number of errors on selecting the proper task category). Another user commented that CTT language might be hard to use without a learning phase and that its approach is better for developing “standard” UIs rather than UIs with customized controls. Another one found it unclear how to modify the model as requested in the test. The last user found using CTTE not intuitive per se, but quite easy to understand after reading the instructions.

## 5 Improved Tool Support

The new version of CTTE, which is also publicly available, addresses some usability issues detected in the user tests. In this section we summarise the main changes carried out.

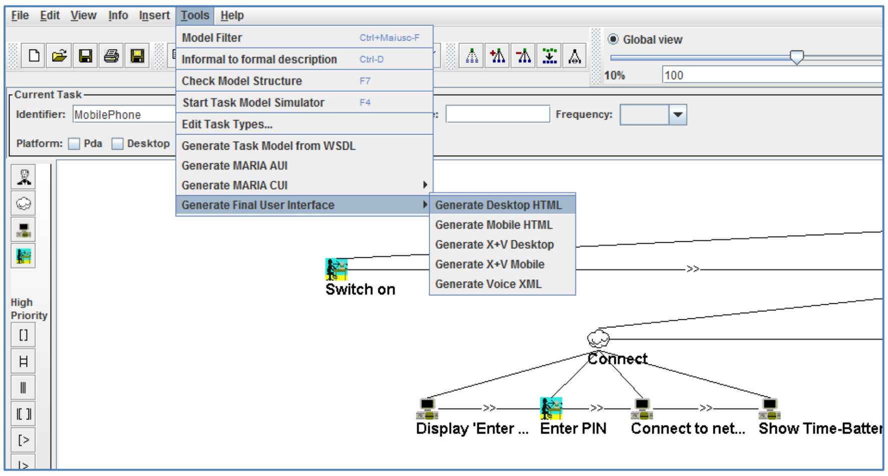

**Fig**. 1 The Activation of various tools in CTTE

In order to derive implementations more easily from task models, we added to CTTE the possibility of automatically generating user interfaces starting from the current task model. This came out from the test with the students: some of them explicitly asked for a tool which provided a preview of the UI automatically derived from the task model. While in the evaluated CTTE version this was not possible, in the new version of the tool, designers can obtain UIs at different abstraction levels. In particular, they can choose (see Figure 1) at what abstraction level (abstract, concrete, implementation) they want to see the corresponding user interface. In the case of concrete descriptions, they can choose on the basis of the target platform (desktop, mobile, vocal, multimodal combining graphical and vocal modalities). The abstract and concrete descriptions are represented by the MARIA language [9]. The implementation languages currently possible are HTML, VoiceXML, X+V, but there are plans to further extend the range. Clearly, what is automatically generated is just a skeleton of the UI, which therefore needs some further editing in order to improve its presentation design. For example, the labels generated are those automatically derived from task and object names and may need some refinements, as well as the choice of the colours, and the size/position of the various graphical elements. However, such early prototypes provide concrete indications of the UI corresponding to the current task model, and can speed up the process of designing usable UIs.

As Figure 1 shows, there are various functionalities that can be activated in CTTE. A recent addition was the automatic generation of CTT models from WSDL descriptions, which is useful where service-based applications are considered.

Another revision, suggested by the user tests, was to add the possibility of editing task names directly on the tree-based graphical representation of the model. In a previous version the tool supported the possibility of editing the name of a task either by activating the window showing all the task properties, or by editing it in a textfield provided in the CTTE main window. Thus, we enabled them to change the task name by just double-clicking on the task name beside the task icon.

As said before, another modification we performed in the tool (which emerged from both the tests) was to remove the automatic check on the task category during the model editing. This check was then left only for those tool features which truly required it (e.g. before activating automatic generation of UIs) or when users explicitly want to check the correctness of the task model. Indeed, while such checks are important when we want to automatically generate UIs starting from the developed task models, in other cases their application may not be required (i.e. during brainstorming about the design of the task model). This was a compromise between the correctness of the task model and the usability of the tool, which led to improved flexibility and better user control (which is in turn typically connected with higher user's satisfaction in the use of the tool).

Another improvement made in the tool (resulting from the test with students) was the possibility to set the (same) temporal operator for more than two tasks. We enabled users to associate one operator to multiple tasks through just a single action, by selecting a set of sibling tasks. This change was introduced to enhance efficiency (by avoiding users making multiple, repetitive actions).

In addition, we introduced the possibility to create and edit task preconditions. The user defines the preconditions by creating a hierarchical representation of the corresponding rule. The user selects the rule group to edit and then selects the Boolean operator to connect the sub-rules or sub-rule groups. In order to edit a rule, the user defines the first and the second operand by selecting either a task object (displayed in a drop-down list) or a constant value (that can be inserted through a text box). Another addition was the possibility to specify postconditions that should be verified after the task performance.

## 7 Conclusions, Future Work, and Acknowledgments

A number of important and more general lessons for model-based tools have stemmed from this experience. Firstly, it is always better to offer a trade-off between the need to present complete information to users and the manageability of this information. Therefore, the idea is to provide a subset of limited information (which should be the most important and frequently used) and then provide further details on request. Moreover, it is advisable to avoid automatic support which can be considered disruptive: correction of the specification is useful when triggered by a user's request, whereas a continuous automatic check can be perceived as intrusive and confusing. The visual tool should be able to support immediate selection and modification of parts of the specification that are logically connected, such as tasks that share the same temporal operator. In addition, it is important to provide some idea of the appearance of the user interface resulting from the model developed.

In conclusion, CTT and its tool represent an interesting case study in the area of visual modelling because of their wide use for various purposes (teaching, industrial applications, research). Here we report on a representative set of issues that have been detected, and discuss how they have been addressed. However, we are aware that various improvements are still possible in various directions. For example, the combined use of graphical and vocal interaction can have interesting applications for facilitating the development of task models, also considering the evolution of vocal technologies in recent years. The wide adoption of touch-based smartphones also stimulates interest in supporting this platform for activities such as editing task models. We also plan further empirical evaluation as well as inspection-based evaluation experiments that exploit methods such as cognitive dimensions.

This work has been partly supported by the SMARCOS Project, http://www.smarcos-project.eu/

## References

1. Cabas Vidani, A., Chittaro, L.:Using a Task Modeling Formalism in the Design of Serious Games for Emergency Medical Procedures. VS-GAMES 2009: 95-102.
2. Jourde, F., Laurillau, Y., Nigay, L.:COMM notation for specifying collaborative and multimodal interactive systems. EICS 2010: 125-134.
3. Luyten, K.,Van den Bergh, J., Vandervelpena, C., Coninx, K.: Designing distributed user interfaces for ambient intelligent environments using models and simulations. Computers & Graphics, Vol.30, No.5, October 2006, pp.702-713.
4. Martinie, C., Palanque, P.A., Winckler, M.: Structuring and Composition Mechanisms to Address Scalability Issues in Task Models. INTERACT (3) 2011: 589-609.
5. Meixner, G., Seissler, M., Breiner, K.: Model-Driven Useware Engineering. In: Model-Driven Development of Advanced User Interfaces, Springer, pp. 1-26, 2011.
6. Mori, G., Paternò, F., Santoro, C.: CTTE: support for developing and analyzing task models for interactive system design. IEEE Transactions on Software Engineering, (2002), 797–813.
7. Paternò, F.: Model-Based Design and Evaluation of Interactive Application. Springer Verlag, ISBN 1-85233-155-0, 1999.
8. Paternò, F.: Tools for Task Modelling: Where we are, Where we are headed. Proceedings TAMODIA 2002, pp.10-17, INFOREC, ISBN 973-8360-01-3, Bucharest, July 2002.
9. Paternò, F., Santoro, C., Spano, L.D.: MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environments. ACM Transactions on Computer-Human Interaction 16(4), 1–30 (2009).
10. Rich, C.: Building Task-Based User Interfaces With ANSI/CEA-2018. IEEE Computer, Vol. 42, No. 9, August 2009.