Chapter #

Computer-Aided Analysis of Cooperative Applications

GIULIO BALLARDIN, CRISTIANO MANCINI, FABIO PATERNO'
CNUCE-C.N.R., Via S.Maria 36
56126, Pisa, Italy
f.paterno@cnuce.cnr.it
Tel : (39) 050 593289 - Fax : (39) 050 904052

Abstract In this paper we discuss how tool-support can be useful for designers who apply task modelling. We focus on how tools can help designers during the development, analysis, and management of task models for cooperative applications. In particular, we introduce how CTTE, a set of tools that we have developed, provides support for the development and analysis of task models of multi-user applications specified in ConcurTaskTrees.

Keywords: Task Models, Cooperative Applications, Tool-support for Design, Formal Methods for HCI.

1. INTRODUCTION

The importance of task models in design, development and evaluation of interactive applications has generally been recognised. However, the use of such task models has strongly been limited by the lack of automatic tools supporting them and a need for such a tool-support has been highlighted also in recent workshops [3]. Indeed, as soon as designers have to address realistic applications they immediately feel the need to have some tools that allow them to analyse the result of their modelling work, to modify it later on, to show it to other people or to use it to implement concrete software-based artefacts that

support the tasks identified. The lack of tool support has been pointed out even for successful approaches such as GOMS [6] and UAN [7].

Tools for supporting environments for the design based on task models would also allow designers to overcome limitations of most current development tools, such as Visual Basic, that allow developers to rapidly obtain user interfaces by direct manipulation techniques but do not provide any suggestion concerning how to structure the user interface and to select the interaction and presentation techniques in such a way as to support effectively the activities indicated in the task model. We can notice that just a few proposals (such as [11], [9]) have been put forwards to overcome such limitations and there are various issues that still need to be solved.

Besides, when tools have been developed to support the design of interactive applications starting from task models (examples are Mobi-D [11] or Trident [2]), they have mainly addressed the design of single user interfaces. Even when they have considered the possibility of different type of users, this was done on the assumption that these types of users do not access the same data at the same time to cooperate through them synchronously. For example, in [11], a tool supporting the design of a user interface for different types of users (sergeant, captain, ..) is described. We too had some experience in designing adaptable interfaces [10]. In that case the problem addressed was to design an interactive application able to support different presentations or interactions depending on current user type and allowing dynamic change of the user type supported directly on request from the user. This gave more flexibility to our approach. However we have recognised that there is an additional issue that in the near future will become always more important: the possibility to support groupware or in other words, applications where multiple users can interact at the same time. This opens a new challenge to model-based approaches that have neglected this issue: how to provide high-level support for designing the specific features of multi-users applications that are becoming more and more common. A need for these new approaches is also motivated by the current toolkits for multi-users interfaces development: even when they address innovative solutions [12], they provide rather low-level constructs to designers and developers.

As described in [5] a groupware system covers three domain specific functions: production, coordination and communication. The *production* space denotes the set of domain objects that model the multi-user elaboration of common artefacts such as documents, or that motivate a common undertaking such as flying an airplane between two places. Typically, shared editors support the production space. The shared artefacts that we can consider are, for instance,

orders and product descriptions. The *coordination* space covers dependencies among activities including temporal relationships between the multi-user activities. For instance, in ERP (Enterprise Resource Planning) applications the coordination should define, for example, how salesmen and clients can communicate and when each of them can perform their tasks. The *communication* space supports person-to-person communication. Email and media spaces are examples of systems designed for supporting computer-mediated communication either asynchronously or synchronously. In particular, coordination seems to be a crucial part in designing cooperative applications.

Generally speaking, we can find applications inherently cooperative such as air traffic control that have always been based on the successful cooperation of different persons and the introduction of software technology allows designers to create environments where this coordination is better supported; and applications, such as ERP applications, that so far have provided low support to cooperations but that could be strongly improved for this purpose thus creating more flexible and interactive environments. We believe that a strong need for redesigning existing applications will arise by the request of better support of cooperation among multiple users.

In this paper we want to report and discuss our experiences in developing an environment that supports the development of task models for cooperative applications specified by the ConcurTaskTrees [8] notation. Such tools are developed within the GUITARE R&D European project (http://giove.cnuce.cnr.it/guitare.html) which aims to improve the design of interactive applications, with particular attention to ERP applications, using task models.

In the paper, we introduce how in ConcurTaskTrees it is possible to describe cooperations among multiple users, next we discuss the tool support required to analyse such cooperations. Then, a more detailed description of our tool, also considering an example of application in the domain of ERP applications is provided and finally some concluding remarks are given.

2. DESCRIBING COOPERATIONS IN CONCURTASKTREES

The ConcurTaskTrees notation [8] allows designers to specify task models hierarchically structured with the possibility to indicate temporal relationships among tasks, objects that they manipulate, how their performance is allocated and so on. Besides, it is possible to specify cooperative applications where different roles are involved. For this purpose there is a part dedicated to the cooperative aspects that allows designers to indicate relationships among activities performed by different users using the same notation.
When developing a task model for cooperative applications, designers first have to identify the roles involved. A role is defined by a set of tasks and a set of relationships among such tasks. Each role has a cardinality indicating the number of users that can be active with that role during an application session. The cardinality can be one, a fixed predefined number or a variable number (in this case the number of users with the considered role will depend on the dynamic evolution of the application). In case of variable number it is possible to indicate a maximum number of users in that role that can be active at the same time.

Then we have a task model according to the ConcurTaskTrees syntax for each role. In addition there is a cooperative part. In the cooperative part we have cooperative tasks structured in a hierarchical manner. A task is cooperative when it implies actions from multiple users. For example, *Negotiating an order* is a cooperative task because it implies actions from both the salesman and the client. These cooperative tasks are decomposed in a hierarchical way until we reach tasks performed by single users. Thus the leaves of the cooperative part are only tasks performed by single users and in addition to the task name there is the indication of the relative role.

A task can occur in different points of the hierarchy. Each occurrence can manipulate either the same objects or different instances of objects of the same class. If a task occurs again in the subtree that it originates then we have a recursive task. If the task that occurs in different places of the task model has a relative subtree then it is sufficient to expand it only once in the task model and it is not required to expand it again in all its occurrences.

The leaves of the cooperative part then are also included in the task model of the corresponding role, thus creating a connection among the two parts. In the single user task tree they can occur in any part. Thus if they are not leaves in the

single user part it means that they can be further refined in subtasks performed by the user of that role.

The purpose of the cooperative part of the overall task model is not only to identify cooperative tasks but also to indicate their temporal relationships, these have the effect of defining additional constraints for the tasks relative to each role. In Figure 1 we provide a simple example to introduce our approach (>> is the sequential operator). Above we have the cooperative part and below the two simple task models associated with the two roles.

If we considered the two roles without the cooperative part we would have at the beginning task1 and task3 enabled to be performed. However the cooperative part adds the additional constraint that task3 can be performed only after task1, thus at the beginning only task1 is enabled.

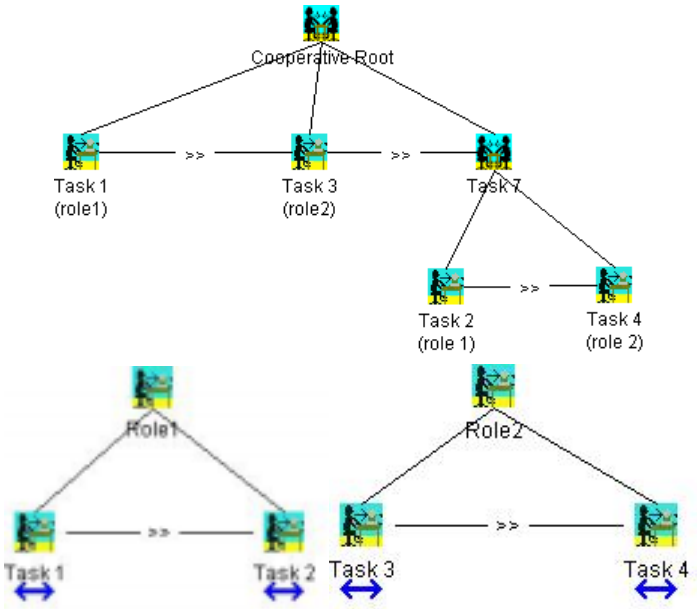

Figure 1. An example of the parts of a simple task model for a cooperative application.

As it is easy to understand the cooperative part allows designers to provide a declarative specification of session managers that control when enabling and disabling the interaction techniques available for each user depending on the state of the cooperation among them.

Designers can need to specify other information. Thus, for each basic cooperative task (a cooperative task which is no longer decomposed into other cooperative tasks), they can specify the following information:

- Task name,
- Roles of users involved,
- names of the sub-tasks which have to be performed by each type of user
- the objects which are manipulated to perform the task, they can be either presentation or application objects, in case of application objects it is possible to specify for each role involved whether it has right access to modify it,
- roles cardinality (how many users for each role are involved in the cooperative task) ,
- some additional informal comments that can be added to further describe the task or some of its main features.

3. TASK MODELLING IN CTTE

We have seen how ConcurTaskTrees allows designers to obtain modular specifications where the various parts can be developed in a consistent way. The resulting specification allows designers to focus on both the behaviour of specific classes of users and the cooperations among the various roles involved clearly indicating their relationships. The tool (CTTE – ConcurTaskTrees Environment) supporting the development of ConcurTaskTrees task models (available at http://giove.cnuce.cnr.it/ctte.html) provides various types of functionalities:

- *to allow designers to input and edit the information requested by the notation*: tasks and their attributes (name, category, type, related objects, …), temporal relationships among tasks and other information;
- *to help designers in improving the layout of their specification*: supporting automatic lining up, movement of levels or subtrees of the task model, cut and paste of pieces of task model, folding and unfolding of subtrees, possibility to add tasks as sibling or as child of the current task, etc.…;
- *to check completeness of the specification*, for example it is possible to automatically detect whether temporal operators among tasks at the same level are missing or whether there is a task with only one child.
- *to save the specification or parts of it in various formats*, the tool also supports functionalities such as inserting other task models in the current

one or saving images containing the task model or part of it so that they can easily included in reports or documents, saving the task model in XML format to easily export it in other environments.

- *to simulate the cooperative task model*, this is useful both in the development of the model and once the result is satisfying, as an interactive documentation of design decisions.

The tool gives also some useful statistical information on the task model: number of levels and of basic tasks, number of iterative and optional tasks, number of tasks for each category (the category of a task indicates how its performance is allocated, to the user or to the application or to their interaction), number of occurrences of each operator used to describe the temporal relationships.

When there are mistakes in the specification detected by the tool it is possible to select one error and then automatically the tool highlights the related part of the specification.

4. AN EXAMPLE OF SESSION USING CTTE

In this section we consider a use case [4] concerning the interaction between a Customer and a Sales Representative to describe how to build the relative cooperative task model associated by using our tool. We first introduce the use case and then we discuss how the tool can be used to develop the related task model.

A Customer visits the Web Page of a Sales Organisation and enters his name and/or Customer ID in order to get in contact with a Sales Representative of the Sales Organisation. The system identifies the Customer and links this customer on-line to the appropriate Sales Representative. Once the connection is established, both the Customer and the Sales Representative can communicate about a shared object via a chosen communication channel, which is in this case a 'chat' program. The shared object is a Draft (for a Sales Order). It is not known beforehand whether this Draft can be converted into a Quotation or an Order, since this depends on the wish of the Customer.

The goal of the Sales Representative is to identify the needs of the Customer, make a sound deal on price and delivery dates and finally be able to convert a

Draft into a Quotation or an Order. The goal of the Customer is to get the correct products just in time at the right place, and for a good price.

The Sales Representative has access to the appropriate data for selling products, while the Customer can only see or enter data that is provided by the sales organisation/or sales person. In fact, the Sales Representative's task of entering the Draft is the basis of the co-operation with the Customer. During the interaction, the Sales Representative and the Customer are totally free in determining the order of entering the data, although only one value can be entered at a time.

The Customer's wishes are the basis for the actual working order, while the Sales Representative has all the 'power' on the task's functionality. When the Customer and Sales Representative agree on the Draft, the Sales Representative can trigger the conversion of the Draft into a Quotation or an Order. There is also a possibility for the Sales Representative to save the Draft temporary for later conversion.

Next to the 'standard' situation of having the Draft as basis of the co-operative interaction, there is also the possibility to interact about a Quotation or an Order as shared object between Sales Representative and Customer. In these cases, the Sales Representative opens these objects. The Sales Representative can only edit the Quotation attributes and convert the Quotation into an Order.

Once the tool is started it is in "single user" modality. In this case we can build only Task Models without cooperative behaviour. By selecting the "Cooperative mode" option in the "Tools" menu we are able to starting a new cooperative Task Model.

In Figure 2 we can see CTTE, in cooperative modality, at the beginning of a session. Initially designers have only the Cooperative Tab Panel associated with the Cooperative part of the global task Model. The designer can either start to model the cooperative part or the task model associated with the roles involved in the application, "Sale Representative" and "Customer" in our example. In the latter case s/he has to select the "New Role" button in the "Commands" panel. We prefer to start to build the task model associated with the two roles of the use case that we consider in our example, and during this developing phase switch to the "Cooperative" part to insert cooperative tasks involving multiple roles whenever we identify a need for their inclusion.

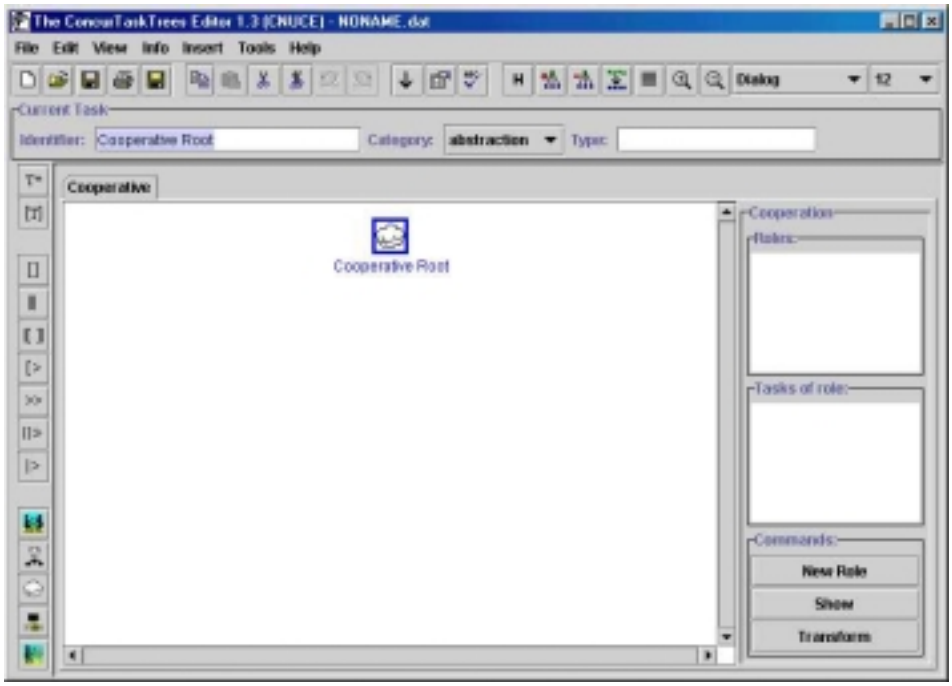

Figure 2. The CTTE tool in Cooperative mode

To explain how to build a generic task model associated with a role we can consider the "Sales Representative" role.

From the textual representation of the use case we identify the main tasks of the role considered and we start to insert them in the tree structure (for example, *Receive message*, *Manage Draft*, *Mange session*, *Manage conversion*). We can insert a new task in the tree structure as sibling, child of parent of the current task selected. We can insert a temporal relationship between task siblings representing the temporal order among them (for example, the salesman transforms a draft int a quotation only after that s/he receives an explicit request from the customer). The current task is identifiable because it is represented with a blue frame around the corresponding icon indicating the category of the task. The insertion can be made by the drag and drop feature from the category icon visible in the left side of the editor main window to a generic task visible in the tree. How the new task is inserted depends on where the cursor is when dropping the icon of the selected task.

It is possible at any time to switch to the view of the task model associated

with another role or the cooperations part. The tool provides standard editing operation: it is possible to copy, cut and paste a task sub tree or a task selected, it is possible to undo and redo operations if the designer makes mistakes. It is also possible to move a single task (mouse left button) or a sub tree (mouse right button) in the work area to enhance the layout of the tree view. It is possible use the "Justify" features, zoom operations or unfold a sub tree to reduce the space used to draw the tree structure when the task model reach big dimensions. For each task, designers can specify some additional information not provided during the development of the tree structure. For example, they can provide a textual description of the task useful to describe more in details the task features, the task type, the task frequency and specify the objects used to perform the task and the relative actions. The tool can specify if a task of a specific role is involved in a cooperative task annotating it with a blue double arrow under the task name. Figure 3 shows the tool during the developing of the task model corresponding to the use case considered.

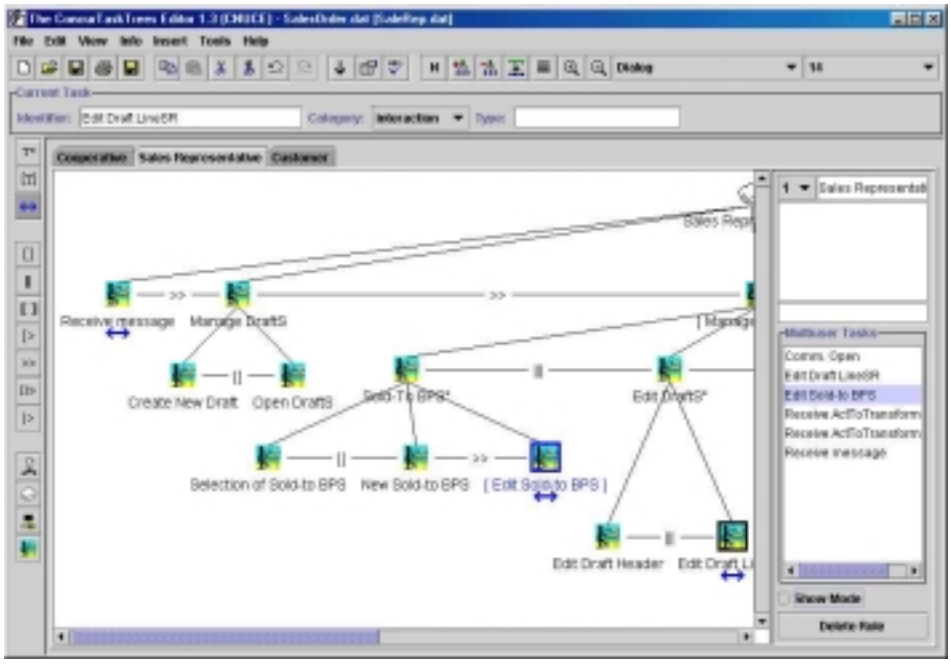

Figure 3. The CTTE tool in an advanced state of task model developing.

To build the cooperative part of the task model we have to select the "Cooperative" tab panel. Once we have identified some tasks part of cooperative

tasks for each role (task with blue double arrow) it is possible to group them with the relative temporal constraints in the cooperative part. To prevent designers from making mistakes during this phase and to help them to find rapidly the task involved in cooperations, the tool shows the list of all tasks involved in cooperation belonging to each role.

Figure 4 shows the task model relative to the cooperative part. The tool allows designers to browse the complete task model. There is the possibility to interactively select a basic task in the cooperative part and then ask an automatic presentation (by the "Show" button) of the single user task model where that task occurs. There is also the automatic check of correctness to identify some mistakes in the Task Model structure. For example, if some temporal operators are missing or if in the cooperative tree designers have not specified all tasks part of a cooperative task that have been indicated in the task models associated with each role (or vice versa). Moreover, some semantic errors in the temporal relationship can be detected: for example, it is not possible to have an iterative task followed by the enabling operator (infinite cycle that would never enable the next task).

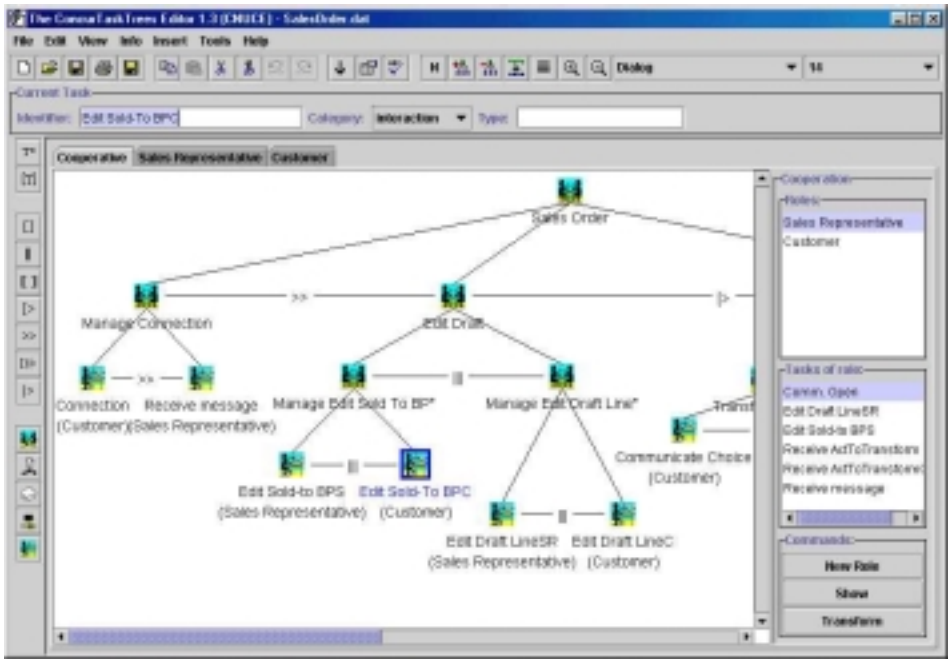

Figure 4. The specification of temporal constraints in the cooperative part.

5. SIMULATING COOPERATIVE TASK MODELS

Some simulators for task models have already been proposed [3]. However, they have mainly been developed for single user task models whereas multi-user application would particularly benefit from them. The resulting behaviour of an interactive cooperative application can be difficult to precisely understand without any support because it is the result of various parts interacting with each other. Thus, some tool support to simulate the behaviour of the cooperative task model can give additional information and allows the designer to check that the specification models the desired behaviour. Such a support can also be useful to discuss with other persons (such as end users) how tasks can be performed in an application so it is also a useful additional interactive documentation.

In our environment, when the simulator is activated there is a panel on the right side listing the tasks enabled. These are the tasks whose preconditions are satisfied and that are ready to be performed, according to the task model. In the central area the designer can interactively select one part of the overall task model and the tool presents the related hierarchical structure with temporal operators. Enabled tasks are highlighted by specific frames.

The designer can interactively select one of the enabled tasks and asks the simulator to perform it, then the simulator updates the state of the task model and shows an updated list of enabled tasks. It is possible to know also what the active tasks are. An active task is a high level task that has at least one subtask performed and has not been completely terminated. In this manner designers can see if the behaviour of the task model built satisfies their expectations. If the behaviour is different with respect to that expected designers can stop the simulation at any time and change the task model structure of a particular role or the cooperative part.

The tool gives also the possibility to store the history of tasks performed during a simulation. This is a high-level indication of a scenario supported by the task model considered. That can be executed again also on a different task model for example to check whether it is able to support it. At any time the designer can stop the simulation and move to the editor to modify the original task model if it has shown some limitations or errors.

In Figure 5 we can see the Task Model Simulation in action. Once simulation is started the list of the enabled tasks appears at the right side of the main window of the tool. So at any time we can see which are the tasks enabled to be performed.

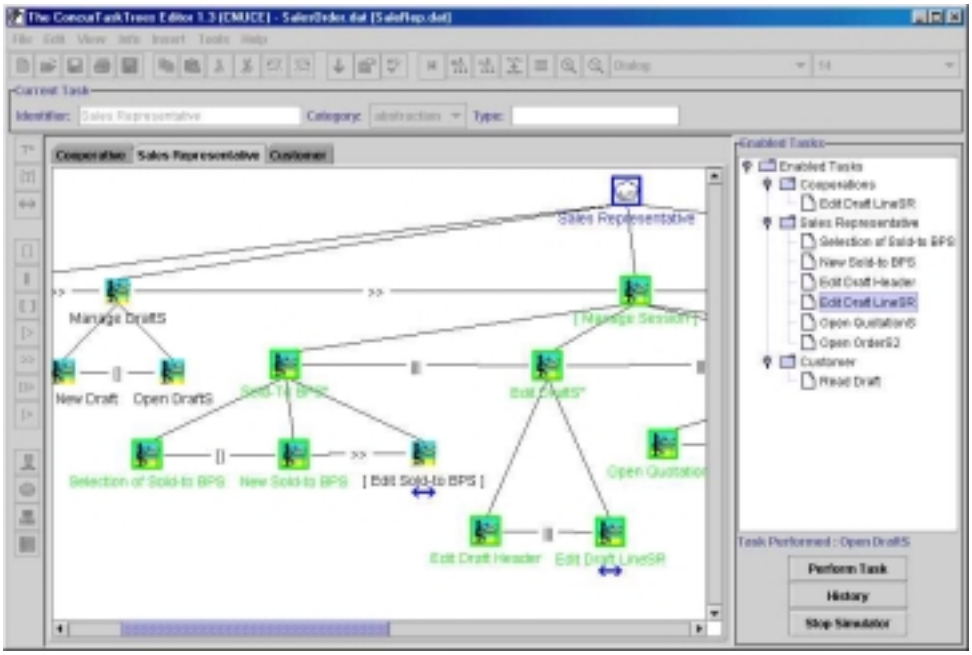

Figure 5. The Task Model Simulation in action.

6. CONCLUSIONS

In this paper we have discussed how it is possible to help designers during the development of task models of cooperative applications and their analysis. In particular, we have focused on how to model cooperative applications in ConcurTaskTrees and the support that can be obtained by CTTE, the relative automatic tool. The tool described (the editor and the simulator of cooperative task models) is publicly downloadable at http://giove.cnuce.cnr.it/ctte.html.

We now plan to use our environment to support the design and development of ERP applications and to develop an empirical study for better evaluating the support that our tool gives to designers and developers.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the GUITARE R&D Esprit Project (http://giove.cnuce.cnr.it/guitare.html) and comments from Marnix Klooster.

REFERENCES

1. Biere, M., Bomsdorf, B., Szwillus G., Specification and Simulation of Task Models with VTMB, *Proceedings CHI'99*, Extended Abstracts, pp.1-2.
2. Bodart F., Hennerbert A., Leheureux J., Vanderdonckt J., A Model-based approach to Presentation: A Continuum from Task Analysis to Prototype, in F.Paternò (ed.), *Interactive Systems: Design, Specification, and Verification*, 1995, Springer Verlag, pp.77-94.
3. Bomsdorf B., Szwillus G., Tool Support for task-Based User Interface Design, *Proceedings CHI'99*, Extended Abstracts, pp.169-170.
4. I. Breedvelt-Schouten, Guitare Use Case, Requirements and Use Case for CSCW-ERP, GUITARE Project Working paper, 1999.
5. G.Calvary, J.Coutaz, L.Nigay, From Single-User Architectural Design to PAC*: a Generic Software Architectural Model for CSCW, *Proceedings CHI'97*, pp. 242-249, 1997
6. Card, S., Moran, T., Newell A., *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum, Hillsdale, N.J., 1983
7. Hartson, R., Gray, P., Temporal Aspects of Tasks in the User Action Notation, *Human Computer Interaction*, Vol.7, pp.1-45.
8. Paternò, F., *Model-Based Design and Evaluation of Interactive Application*. Springer Verlag, ISBN 1-85233-155-0, 1999.
9. Paternò F., Breedvelt-Schouten I., deKonig N., Deriving Presentations from Task Models *Proceedings EHCI'98*, Creete, Kluwiert Publisher, September 1998.
10. Paterno' F., Mancini C., Developing Adaptable Hypermedia, *Proceedings IUI'99*, pp. 163-170, January '99, ACM Press.
11. Puerta A., Cheng E., Ou T., Min J., MOBILE: User-Centred Interface Building, *Proceedings CHI'99*, pp. 426-433, 1999.
12. M. Roseman, S. Greenberg, Building Real-Time Groupware with GroupKit, A Gropware Toolkit; *ACM Transaction on Computer-Human Interaction*, Vol 3 N°1 March 1996; pp. 66-106.
13. van der Veer G., Lenting B., Bergevoet B., GTA: Groupware task analysis - Modelling complexity, *Acta Psychologica*, 91, pp.297-322, 1996.