Formal Models for Cooperative Tasks: Concepts and an Application for En-Route Air Traffic Control

F. Paternò [(1)], C. Santoro [(1)], S. Tahmassebi [(2)]

[(1)] CNUCE-C.N.R., Pisa, Italy
[(2)] CENA, Toulouse, France

Abstract. This paper presents a proposal for specifying task models for cooperative applications that allow designers to describe the relationships between the activities performed by various users involved in cooperative environments. To this end we extend the ConcurTaskTree notation so that new information useful for describing complex cooperative applications can be clearly specified. An example of application to describe En-Route Air Traffic Control (ATC) is given to illustrate and clarify our approach.

1 Introduction

Formal methods are based on the use of notations with a precise semantics that allow designers to clarify many aspects, remove ambiguities in their specifications, and develop rigorous reasoning about the properties of these specifications. However, both learning and applying formal methods is time-consuming. We believe that these costs are motivated when designers consider applications where a bad design can generate heavy consequences.

Safety critical applications [6] are an interesting application area where the current user interfaces need to be improved, as demonstrated by several studies which show how most accidents are predominantly due to human errors and the need to improve their analysis [4]. It is thus important to develop a formal analysis of user interfaces for these applications [5] in order to have a full understanding of the possible effects of user interactions.

One of the most interesting results of the application of formal methods in the HCI field [9] is that precise descriptions of task models [7, 10, 13] can be given. Thus it is important to have a rich set of temporal operators and to define a precise semantics which can explain the behaviour described when these operators are composed in complex expressions.

In the European MEFISTO Project we aim to develop formal task models to improve the analysis and the design of safety critical interactive applications. We started by considering a case study consisting of an application for air traffic control during the en-route phase.

We soon realised that besides being a safety-critical application because an error caused by the controller can have serious consequences, it is also a highly cooperative application. In fact, for each sector there is one strategic controller and one executive controller, who communicate with each other and, in the application currently used, the executive controller communicates with the pilots, and the strategic controller communicates with the strategic controllers of the other neighbouring sectors.
This raises the need for an integrated analysis of usability, safety and cooperative aspects in the design of user interfaces for these applications. In fact, many of the possible safety and usability problems are related to how the various users cooperate with each other to reach common goals.

In the design of complex co-operative environments more and more attention is being paid to the horizontal mechanism of co-ordination between different roles. This calls for consequently more innovative tools and notations to describe these interactions. ConcurTaskTrees [11] is a notation that supplies a hierarchical graphical notation providing a precise semantics for describing concurrent task models of every role. However, in the previous version, it does not allow designers to specify explicitly how the cooperation among different users is performed, since each task model is related to only one user. When we developed the task model for the different types of users we thus realised that our notation provides good representations of task models of single users but it does not highlight sufficiently the co-operative aspects which are relevant for applications such as the ones considered in this report. This is a limitation for many approaches to task modelling, apart a few exceptions such as GTA [13].

We thus decided to develop an extension to our notation which allows designers to describe these co-operative aspects, yet still maintaining compatibility with the other parts of the ConcurTaskTrees specification that describe the task model associated with each user role. We aim to obtain a structured approach to the design of cooperative applications taking into account relevant aspects of this class of applications [1, 3]. More specifically the goals of the work presented in the paper are:

- *to describe a new proposal for specifying cooperative task models* based on the ConcurTaskTrees notation previously developed which has proved to be a powerful approach to describe task models but which was limited to address single users applications;
- *to show an application of the new proposal to a realistic case study*, this is important to understand the scalability of the approach which is a key element if it has to be used in analysing and designing real applications.

In this paper we first discuss the possible representations of temporal relationships between tasks performed by different users, and motivate our choice. Then we give the reader an overview of the main features of the case study considered. We

describe the associated task model and indicate the relevant information that should be provided for describing the cooperative relationships. Finally, some concluding remarks and indications for future works are given.

2 How to specify cooperative task models

Task models for cooperative applications need to integrate different aspects. Designers can specify models describing the tasks from the viewpoint of different users, and then they have to indicate their relationships. This allows designers to analyse and develop task models from different angles and to check consistency and completeness. The viewpoints that we apply in the new version of ConcurTaskTrees are essentially of two types: the first one is the viewpoint of individual users, the second refers to the relationships among the tasks of various users in reaching global objectives. For example in the ATC application we consider below there are many *subjective* views (the strategic controller's view, the executive controller's view, the pilot's view), and only one *objective* view which gives an overall description of the working environment including communication between users.

We define a cooperative task as a task that requires activities from two or more users for its performance. After a user has performed a subtask belonging to a cooperative task, some reaction by one or more users is expected and necessary in order to complete the task. For example, the handling of the transfer of one flight to a new sector or the management of the conflict among two flights are two cooperative tasks since they require activities by more than one user. On the contrary, annotating one flight strip is not a cooperative task, according to our definition, because only one user performs it even though it might be a subtask of a cooperative task.

Note that in a cooperative environment, *all* the activities performed by the users obviously aim to eventually achieve the global goals. Nevertheless, we prefer to define «cooperative» only those tasks that strictly define the structure of cooperation, so that the roles users play with respect to the whole cooperative activity can be highlighted.

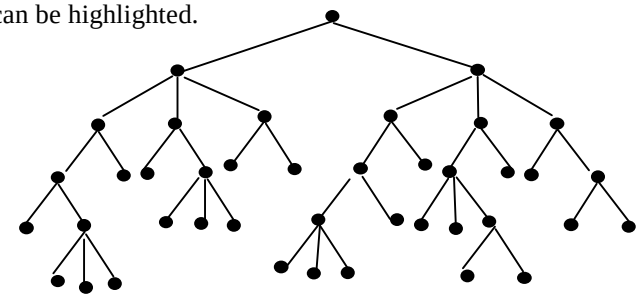


Fig. 1. The monolithic solution for cooperative task models

In order to extend the ConcurTaskTrees notation to explicitly providing representations of cooperative aspects there are at least three solutions:

- *A monolithic solution*, to develop directly one global task model including cooperative and single users tasks (see Figure 1). Note that in this picture (as in the next two pictures) the parent-child relationship describes the hierarchical decomposition of the task in its sub-tasks;
- *A graph-oriented solution*, to join together the task tree of each role involved in a single structure in which any co-related actions or sub-tasks (performed by distinct users) are connected by lines or arcs with operators showing existing temporal relationships. For example, it should be possible to show a task of user1 activating a task of user2 by a line connecting the two tasks and the indication of the related temporal relationship associated with it (see Figure 2).

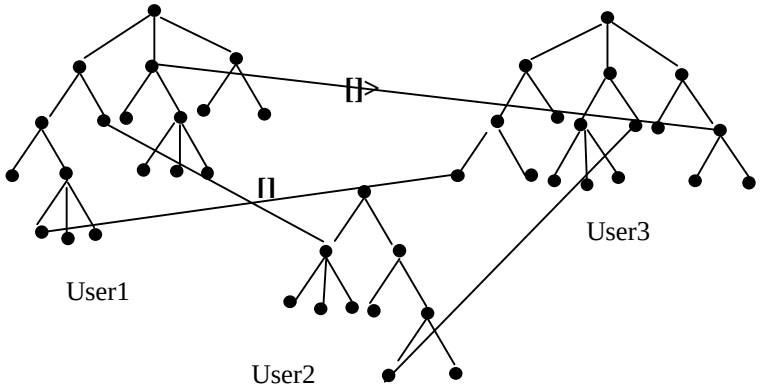


Fig. 2. A possible representation of temporal relationships between tasks of different users

- *An additional cooperative task solution*, to add another tree (the *Co-operative* tree) which shows the temporal relationship between tasks belonging to different users (see Figure 3). For each basic task the user who performs it is indicated. Then we still maintain for each role a tree indicating the related tasks.

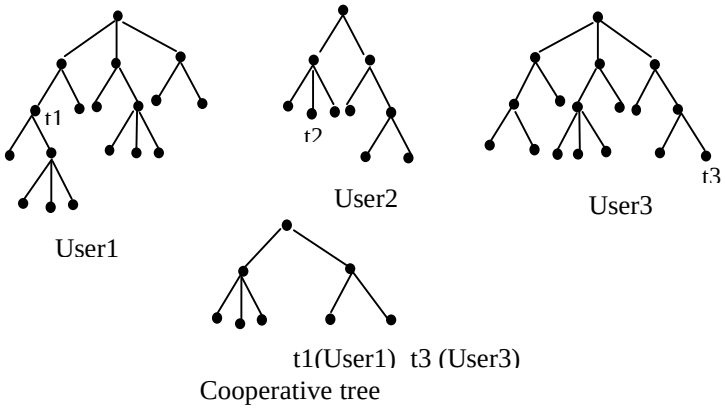
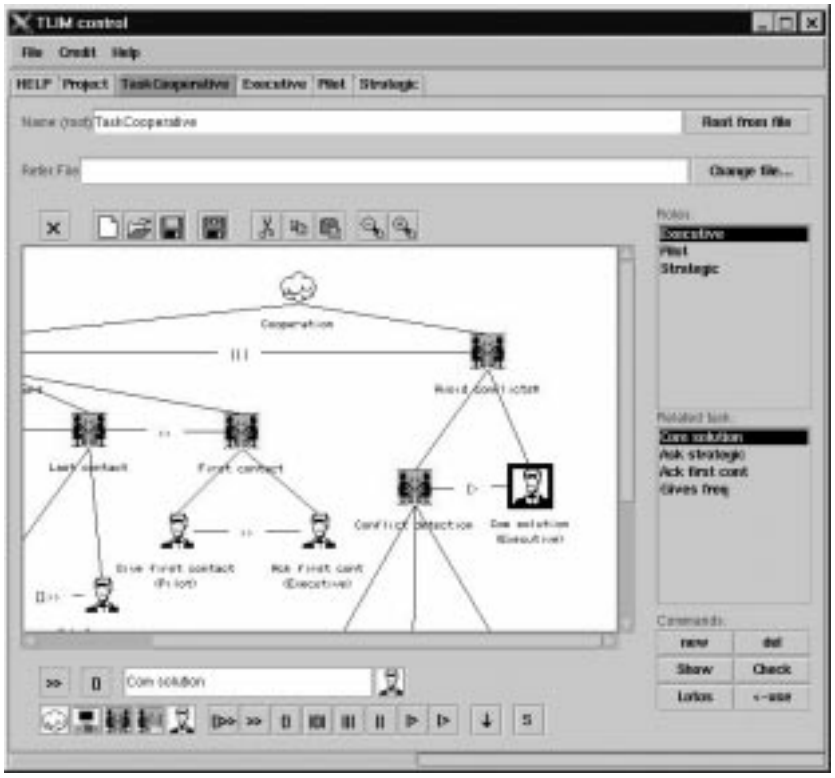


Fig. 3. Another possible representation of co-operative tasks

The first solution can easily generate task models that are too complex to interpret. Although the second solution is immediate, it does not seem to be the most effective especially when there is a high number of relations. In fact, as shown in Figure 2, the understanding of the overall behaviour might be difficult because the intersecting links could generate a complex graph.
We have decided to adopt the third solution (i.e. an additional cooperative task) because it enables designers to edit the cooperative part of the task model separately. This solution highlights the communicative aspects (information and/or activities) from the viewpoint of the tasks that involve cooperation among two or more users, their main features and their relationships.

3 Tool support for cooperative task models

In the specification of the co-operative aspects we still use a hierarchical representation with operators to describe temporal relationships among tasks at the same abstraction level. Thus we obtain an additional task tree whose main purpose is to define the relationships of the task trees associated with each type of user.
In this task tree we decompose the cooperative tasks until we obtain tasks performed



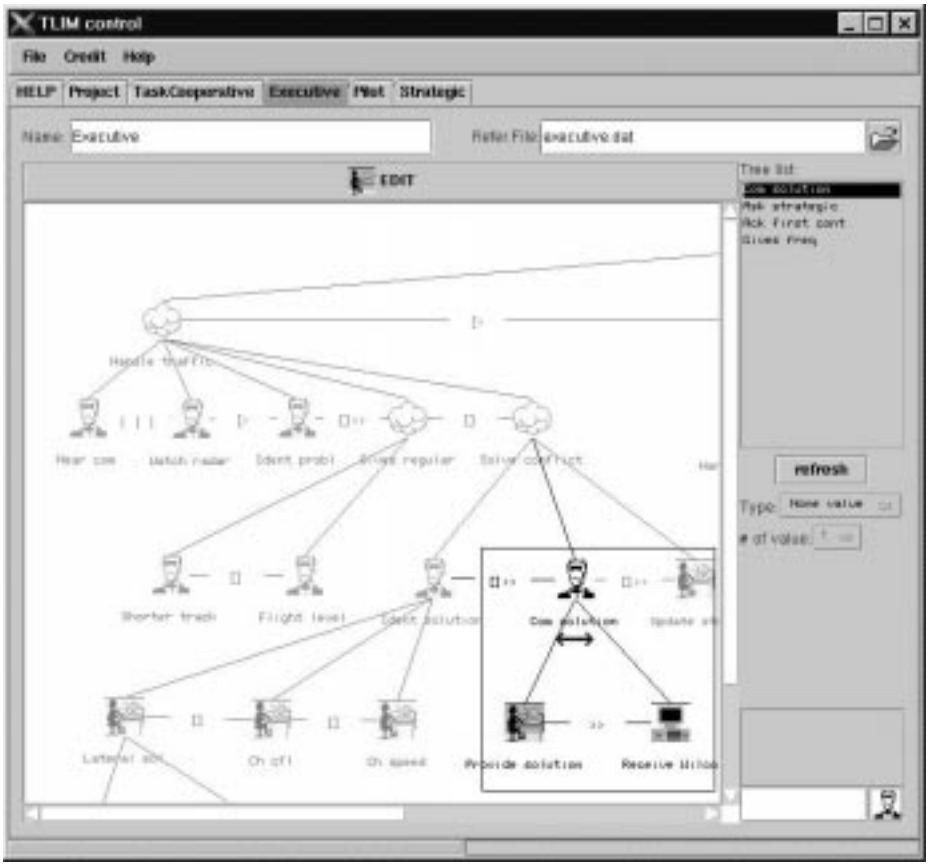
by a single user. If such tasks can be further decomposed into subtasks performed by

that user, the description of this decomposition is given in the task tree associated with the related user.

Fig. 4. The tool supporting analysis and editing of cooperative task models.

The leaves of the cooperative task tree can be either *basic* or *high level* tasks in the tree of the related user. In the first case, leaves in the cooperative tree remain leaves in the trees of the related users, whereas in the latter case the leaves in the cooperative tree will be further decomposed in the tree of the related user.

We are implementing a tool that supports the editing of cooperative task models in the third solution. When the user selects a basic task in the model of the cooperative part (see Figure 4, the *Com solution* task) this triggers another window (Figure 5).



The new window shows the decomposition of the selected task in the task tree of the corresponding user, thus highlighting their relationships.

Fig. 5. An example of tool-supported analysis of the relationships between in the task model

In order to develop and analyse a cooperative task model we can follow two main approaches:

- a *top-down* approach, in which —starting from the overall activity of the system— we should derive the activities of the single roles and in what extent every role contributes to carry out the general activity
- a *bottom-up* approach, where the first analysis of every role's activity with the consequent initial task allocation helps to achieve a more modular comprehension of the overall behaviour in the cooperative application.

We preferred the second one in our case study because it allowed us a more structured and simpler way to achieve the specification of the task model for the cooperative application considered. Anyway it is worth to point out that the development of the specification was not a one-step process. It was an iterative process because once we had defined the specification of all the task trees (one tree for every role, and the cooperative tree) we had to check and modify it until the complete and correct specification was achieved. For example, if analysing whether all the activities had been described in the cooperative tree, we discovered that some were missing then we had to add them in the cooperative tree and then to analyse how every role takes part in them. Another point to note is that this approach forces the designer to check the possible inconsistencies or ambiguities inside the specification, so improving the correctness of the specification.

Designers need to specify other information apart from the roles that execute the subtask of the cooperative tasks. Thus, for each basic cooperative task (a cooperative task which is no longer decomposed into other cooperative tasks), we allow them to specify the following information:

1. task name,
2. roles of users involved,
3. names of the sub-tasks which have to be performed by each type of user,
4. the interaction media used to communicate,
5. type of communication protocol (for example, synchronous or asynchronous, point-to-point or broadcast), this aspect is often related to the media chosen,
6. the objects which are manipulated to perform the task,
7. roles cardinality (how many users for each role are involved in the cooperative task),
8. some additional informal comments that can be added to further describe the task or some of its main features.

4 The case study considered

We have considered an application for air traffic control during the en-route phase. The air space is divided into sectors. Two controllers control each sector: the executive and the strategic. They both perform surveillance and system updating tasks. Also, the executive controller is responsible above all for direct communication with pilots (by radio), the strategic controller for communication with the strategic controllers of other sectors.

Fig. 6. The contextual environment of a controller

The environment of a (French) controller (see Figure 6) includes radar screen, paper strips, and a touch input device which allows controllers to update in the ground system some flight data such as the time, level and track. The ground system generates and prints the paper strips, displays radar data, and manages flight plans. The paper strips are printed and delivered to a given control position about ten minutes before the flight enters the sector handled by that position. The main goal of this application is safety and regularity of flights (shorter time, less fuel consumption). In this context the main safety problem is to avoid flight conflicts, when two or more aircraft are in the same place at the same time.

5 The task model of the strategic controller

Below is an informal list of what a strategic controller typically does when only radio communication with pilots is supported:

- They receive information on paper strips about entering aircraft.
- They check possible conflicts with other aircraft in the air. If they detect any they inform the executive controller.
- When a flight is changing sector if problems are detected then they phone the strategic controller of the other sector and negotiate the optimal flight parameters to enter in the new sector (change parameters: level, track, etc.). When there is a change in sector the sender controller can change parameters and update the system, the receiver controller takes the strips, checks the colour and gives them to the executive.
- They can perform surveillance tasks: check conflicts and communicate with the executive for modifications.
- They can update the system at any time and not only when the aircraft is leaving a sector.

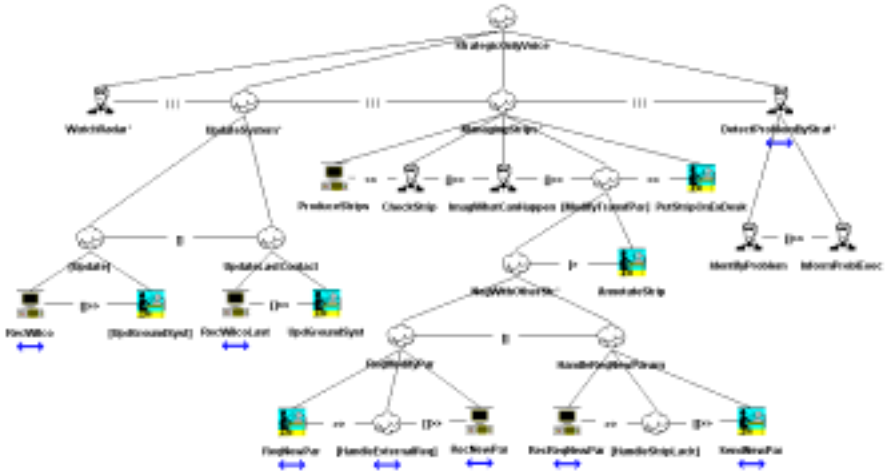


Fig. 7. The task model of the strategic controller

Figure 7 describes the activities of a strategic controller. There is a set of iterative activities (this is described by the iterative operator, *). At a high level of abstraction we can identify four main activities: watching the radar for surveillance purposes, updating the ground systems with the modifications of flight plans performed, managing strips, and detecting problems.

The concurrent activities (composed by the ||| operator) can be performed without any constraints on the number of times (thus these tasks are iterative). In the specification in Figure 7 we show only the first levels of the specification thus when an abstract task is on the bottom of the tree it means that it is further decomposed in the complete specification

Update system is decomposed into two cases: when the pilot communicates that the last contact will be performed then the system has always to be updated whereas in the other cases the updating of the system is optional (optional tasks are indicated with the name in squared brackets).

Managing strips means to check the paper strip which is generated by the system and contains information on a flight arriving into the sector. By considering the flight plan described in the paper strip the strategic controller has to imagine what can happen during the temporal evolution. Then there is an optional task. The option depends on whether there is the need to modify some flight parameter. If some parameters have to be modified this requires a negotiation with the strategic controller of the sector from where the flight is coming. In case of modification of parameters the strip is annotated. Finally the strip is located on the desk of the executive controller.

When a problem is detected the strategic controller informs the executive controller. The problem can be identified either watching the radar or because of the strips. The problem with the strips can be because the system failed to generate them either in

the sector of the controller considered or in another sector (and thus there is a request of information from the related controller).

6 The task model of the executive controller

Below is an informal list of the main tasks for executive controllers:

- They receive flight strips and deduce the flight evolution;
- They receive the first contact request from the pilot with its flight parameters;
- In the first contact task the following information is indicated: identification, flight level. The executive checks if they are okay and gives clearance, otherwise s/he changes the info on strips;
- They give to the aircraft the frequency for the next sector;
- They give the optimal flight level;
- They control the aircraft and detect any possible conflicts (if conflicts are detected then they have to create a 3D separation whose parameters depend on the altitude).

Fig. 8. The task model of the executive controller

The task model of the executive is first decomposed into two main activities*:* handling traffic inside the secto*r* and handling change of sector of one flight. The first activity is decomposed in watching the radar for surveillance purposes and handling the air traffic (giving regularity to the flights and avoiding conflicts).
Once the executive controller identifies a problem which may require either giving more regularity to one flight (by giving a shorter track or changing a flight level) or solving a conflict then s/he has to identify a solution (that in the case of conflict can be a lateral solution, changing heading or beacon, or a vertical solution, changing flight level, or a longitudinal solution, changing speed) and finally to communicate it.

HandleChanging sector means either that a flight is leaving the sector or that a flight is entering into the sector.
In case of leaving flights, the executive controller gives the frequency of the next sector, receives the answer from the pilot and then removes the strip of the leaving flight. When there are entering flights the executive controller receives the first contact from the pilot, replies and then has to manage the related strip. Finally he can detect whether some flight parameters do not comply with the flight plan (optional task).
We recall that the priority order among operators is: choice > parallel composition > disabling > enabling, where choice is the "[]" operator, enabling is the ">>" operator and parallel composition can be either completely interleaving among tasks ("|||" operator), or interleaving with synchronisation on some actions.

7 The task model of the pilot

The task model of the pilot is very simplified and only considers those parts that are strictly connected to the tasks performed by the controllers. The pilot's normal behaviour can be interrupted by the *RequireExecOperation* or *Changing sector* tasks. In the ordinary course of events the pilot receives messages from the controllers, follows the flight plan and monitors the equipment.
RequireExecOperation means that the pilot sends a request to the executive controller for performing an operation. Then he waits for the answer and finally he sends either a command indicating the performance of the operation or the inability to perform it. When changing the sector the pilot first receives the frequency of the new sector from the controller of the old sector and then he provides the first contact to the controller of the new sector. In some cases after the first contact the executive can ask some information to the pilot.

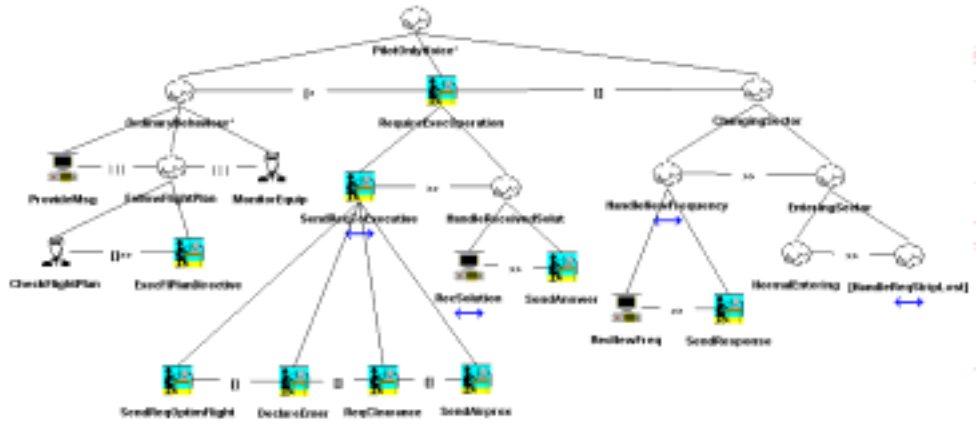


Fig. 9. The task model of the pilot

8 The specification of the cooperative tasks

We have defined above a *co-operative* task as a task which requires actions from two or more users for its performance. For example, the *Give Solution* task (see Figure 10) requires actions from the executive controller who sends the solution and the pilot who receives it.

In the specification in Figure 10 we can see that there are two main cooperative activities: one related to the change of sector of one flight and the other related to avoiding conflicts inside a sector.

The former can be subdivided into other cooperative tasks: handling last contact and first contact. After a first contact of one flight it is possible to activate the change of sector of another flight and the last contact of that flight concurrently.

The *HandleTrafficInSector* task can require cooperation among the executive controller, the strategic controller and the pilot or only between the two controllers. The cooperation can be motivated by the goal to optimise the flight (more regularity or less fuel consumption) or to solve a new problem.

The *ChangingSector* task can require also cooperation among strategic controllers associated with different sectors.

Fig. 10. The Cooperative tree of the example

The tasks that are performed to complete a cooperative task are indicated by the double arrow symbol in the related single user task model, whereas in the specification of the cooperative part there is one new type of icon (with two persons)

which indicates cooperative tasks. In Figure 10 we show only the higher levels of the specification of the cooperations. When leaf in the tree is associated with a cooperative task it means that it will be further decomposed in the complete specification.

9 Templates for basic cooperative tasks

In addition, for each task in the cooperative part which cannot be further decomposed into other cooperative tasks we allow designers to 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, type of communication protocol (for example, synchronous, asynchronous and broadcast), roles cardinality, the media used to interact and communicate, the logical objects that the task needs to manipulate and some additional informal comments which can be added to further describe the task or some of its main features.

For example the *Send first* cooperative task which is part of the *Changing sector* co-operative task is described as:
task name: Send First; *roles involved*: pilot, executive controller; *subtasks performed by the users involved*: give first contact (pilot), acknowledge first contact (executive controller); *interaction media*: radiotelephony; *communication protocol*: asynchronous; role cardinality is one executive-many pilots with radiotelephony; *objects*: the conceptual objects used in this task are: frequency, sector and flight (see Table 4).

The *Send first* task involves two roles: the pilot and the executive controller. The pilot calls the executive controller of the entering sector. The controller replies with an acknowledgement message. They perform the *Give first contact* and *Ack first contact* subtasks respectively in order to complete the cooperative task. The communication flow is bidirectional, and it can be realised by radiotelephony. Because all the pilots crossing a sector share the same frequency the communication from the executive controller to the pilot is from *one* executive controller to *n* pilots in the radiotelephony case. We indicate an asynchronous communication protocol as we assume that after that the controller has given the frequency s/he can still perform other tasks before receiving an answer from the pilot.
We use a tabular representation to clearly indicate the information associated with a co-operative task as mentioned above. For example, the information concerning some of the cooperative tasks in the cooperative tree in Figure 10 are shown in the following tables.

T1	*Task name = Send Last Contact*
Roles and subtasks	*Executive*: SendNewFreq, RecWilcoLast *Pilot*: HandleNewFrequency *Strategic*: RecWilcoLast
Objects	Frequency, Sector, Flight
Interaction Media	RadioTelephony (VHF or HF)
Protocol	Asynchronous - Broadcast
Role Cardinality	Pilot : Executive : Strategic (N,1, 1)
[Comments]	The executive controller of the exiting sector gives the new frequency to the pilot who provides an answer to the controllers.

Table 1. The table for *Send Last Contact* task

T2	*Task name = Optimise flight*
Roles and subtasks	*Pilot*: SendReqToExecutive *Executive:*: RecReqFromPilot
Objects	Flight level, Speed, Heading
Interaction Media	Telephone
Protocol	Asynchronous - Broadcast
Role Cardinality	Pilot : Executive (N,1)
[Comments]	The pilot asks for some optimisation of the route of the flight

Table 2. The table for *Optimise flight* task

T3	*Task name = ModifyTransPar*
Roles and subtasks	*Strategic*$_{i}$: ReqNewPar, RecNewPar *Strategic*$_{(i+1)}$: RecReqNewPar, SendNewPar
Objects	Frequency, Sector, Flight
Interaction Media	Telephone
Protocol	Synchronous - Point to Point
Role Cardinality	Strategic$_{i}$↔ Strategic$_{(i+1)}$ (1,1)
[Comments]	The strategic controller of one sector interacts with the strategic controller of a neighbouring sector in order to negotiate the parameters of one flight which is changing sector

Table 3. The table for *ModifyTransPar* task

10 Conclusions

In this work we have presented a new approach to modelling tasks in cooperative applications and we have shown its application to a case study in the air traffic control field. We have identified various benefits of the use of formal task models:

- An understanding of how the activities in the current work context of the application considered are carried out;
- The possibility of analysing a cooperative activity from different viewpoints (the *subjective* points of view of each role involved in the cooperation and the *objective* point of view of the overall activity);
- A useful aid to the identification of what the "shared" (in some sense "critical") objects involved in the cooperation are.

Future work will be dedicated to consider the following aspects:

- The use of task models for analysing and evaluating the different organisation of activities determined in the application considered by the introduction of new technologies or by different design choices. For example, the introduction of a data link communication between controllers and pilots;
- In order to consider more safety-critical applications we plan to integrate the approach presented with Hazop-like techniques. The goal is to combine the advantages of the first one (to clearly specify the cooperations occurring in an application) with those of the second one (to support identification of possible deviations with respect to the design intent, during the cooperations, which can generate hazardous situations).

Acknowledgements

This work is supported by the MEFISTO (http://giove.cnuce.cnr.it/mefisto.html) European Esprit Reactive Long Term Research Project (N.24963).

References

1. Calvary, G., Coutaz, J., Nigay, L.: From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCW, pp. 242-249, ACM Press, Proceedings CHI'97.

2. Druart, J., Novales, L.: En route Control Standards. Mefisto Report WP1-4, November 1997.

3. Dix, A.: In: Beale R. (eds.): Remote Cooperation, Springer Verlag 1996.

4. Field, R., Wright, P., Harrison, M.: A task centred approach to analysing human error tolerance requirements. IEEE Press, Proceedings RE'95, York.

5. Johnson C.W.: The Formal Analysis of Human-Computer Interaction During Accident Investigations. In: Cockton, G., Draper, S., Weir,G. (eds.): People and Computers, IX, pp. 285-300, Cambridge Uni. Press, 1994.

6. Levenson, N.: Safeware, System Safety and Computers. Addison Wesley Publishing Company, 1995.

7. Markopoulos, P., Johnson, P., Rowson, J.: Formal Aspects of Task-based Design., pp. 209-224. Springer Verlag, Proceedings DSV-IS'97.

8. Palanque, P., Bastide, R., Paternò, F.: Formal Specification as a Tool for Objective Assessment of Safety Critical Systems, pp. 323-330. Chapman&Hall, Proceedings Interact'97.

9. Palanque, P., Paternò, F. (eds.): Formal Methods in Human-Computer Interaction. Springer Verlag , FACIT Series, ISBN 3-540-76158-6, 1997.

10. Paternò, F.: Understanding Task Model and User Interface Architecture Relationships, CNUCE Internal Report, December 1997.

11. Paternò F., Mancini C., Meniconi, S.: ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, pp. 362-369, Proceedings Interact'97.

12. Tahmassebi, S.: Controller Pilot Data Link Communication (CPDLC) description, Mefisto Report WP1-1, November 1997.

13. Van der Veer, G., Lenting, B., Bergevoet, B.: GTA: Groupware task analysis - Modelling complexity, pp. 297-322. Acta Psychologica, 91, 1996.