Using Information in Task Models to Support Design of Interactive Safety-Critical Applications

Fabio Paternò
CNUCE-C.N.R. Institute
Via V.Alfieri, 1
56010 Ghezzano, Pisa, Italy
+39 050 3153066
f.paterno@cnuce.cnr.it

Vincenzo Sabbatino
Alenia Marconi Systems
Via Tiburtina Km 12,400, Rome, Italy
00131 Rome, Italy
+39 06 41 50 31 95
svin@lti.alenia.it

Carmen Santoro
CNUCE-C.N.R. Institute
Via V.Alfieri, 1
56010 Ghezzano, Pisa, Italy
+39 050 588 3153053
c.santoro@cnuce.cnr.it

ABSTRACT

The use of models has entered into current practice when developing various types of software product. However, there is a lack of methods able to use the information contained in relevant models concerning human-computer interaction for supporting the design and development of user interfaces. In this paper, we propose a method for using information contained in formally represented task models in order to support the design of interactive applications, with particular attention to those applications where both usability and safety are the main concern. Examples taken from our experience in a case study from the domain of Air Traffic Control are introduced and further discussed to explain how the method can be applied.

Keywords

Model-based design of user interfaces, task models, interactive safety-critical applications.

1. INTRODUCTION

Model-based design of interactive applications is a research area concerning approaches aiming at identifying models able to support design, development, and evaluation of interactive applications. Such models highlight important aspects that should be taken into account by designers. To support the design of visual interactive applications various types of models have proved to be useful, such as user, context, and task models. Such models were not supported in the early model-based approaches, such as UIDE [4] and HUMANOID [8], that used lower level abstractions.

In addition, even if the use of task analysis and modelling has subsequently been applied in the design of interactive applications, there is still a lack of engineering approaches to the use of task models. An engineering approach should require at least:

- Use of flexible and expressive notations with precise semantics able to represent the different ways to perform tasks and the many possible temporal and semantic relationships among them;
- Systematic methods able to indicate how to use the information contained in the task model for supporting the design and evaluation of the user interface;
- Availability of automatic tools able to make the development and analysis of such task models more efficient.

Recently interest in solving this issue has arisen also because the identification of systematic methods gives the possibility to develop automatic environment that can support the use of models and related methods. Something similar already occurred in the development of software systems with the success of UML [2]. However, despite the nine notations that UML provides it is clear that it is inadequate to support the design of user interfaces because these notations have mainly been developed to support the development of the internal structure of a software system. Of course, they can be used to design and specify user interfaces but they are not effective for this purpose. Recently, few contributions have been put forward in the area of model-based design of interactive applications [1, 7] and there are many issues yet to be resolved. The HCI group at CNUCE is involved in a European project (MEFISTO) where task models are considered for supporting design of interactive safety-critical applications, with particular attention to air traffic control. This is an area where interesting studies on the application of novel interaction techniques have been carried out [3, 5]. The project involves industrial partners such as Alenia Marconi Systems that have long tradition in developing this type of application. More specifically, in MEFISTO we want to understand to what extent the use of rigorous techniques, developed in the area of formal methods, can help in the design of interactive safety-critical applications. To this end, we pay particular attention to the use of task models to support such a design and analyse how possible user deviations during task performance can have an impact on safety.

In this paper, we briefly describe the modelling work developed in the case study, next we discuss some general criteria to use information contained in task models to support user interface design, and then we show how they have been applied in parts of the case study we have developed. Finally, we further discuss a set of criteria that can be the core for guidelines to support the design of interactive safety-critical applications.

2. DEVELOPING THE TASK MODEL

The task models developed have been represented using the ConcurTaskTrees notation [6]. The purpose of a task model specified in ConcurTaskTrees is to provide a description of how

the activities should be performed in order to reach the user's goals. Such activities are described at different abstraction levels in a hierarchical manner, represented graphically in a tree-like format. In contrast to previous approaches, ConcurTaskTrees provides a rich set of operators with a precise semantics to describe the temporal relationships among such tasks. The notation gives also the possibility to use icons or geometrical shapes to indicate how the performance of the tasks is allocated: only to the user, only to the application, interaction between user and application, abstract tasks (which means that they have subtasks allocated differently). For each task it is possible to provide additional information including the objects (for both the user interface and the application) manipulated.

Figure 1: The Fiumicino Control Tower

We have considered as case study the air traffic control in an aerodrome. The increasing air traffic in the last years has highlighted the need for better support in these areas where aircraft, and other vehicles, are particularly concentrated. This problem is more evident in case of bad atmospheric conditions. The development of the task model of the current system has been carried out with the support of information gathered in different ways. We have visited various times the control tower of the Fiumicino airport in Rome (the photo in Figure 1 was taken during one of these visits), followed by interviews with controllers. We have had various meetings with a team of Air Traffic Control (ATC) systems developers who have long experience in developing such applications and are involved in the development of the new prototype.

The current user interface available to controllers in the tower is composed of a set of rather rudimentary devices (see Figure 1): the windows where they observe the traffic, the radio for communicating with pilots, paper strips containing flight data, and radar screen with basic information concerning flights. We developed the task model for the existing application considering in particular the ground controller that is in charge of handling control of traffic between the gates and the runway and tower controller that handles the take-off and landing. Modelling the existing work methods was useful to reach a deeper understanding of what the activities to support are, what relationships occur among such activities and possible problems and limitations. At the highest level of the ground's task model three main tasks are recognisable: handling the paper strips, maintaining and updating the picture of the current/future traffic situation and driving the traffic under his/her responsibility.

If we analyse the task model of the Tower controller we note that it presents relevant similarities with the ground controller despite the fact that they manage aircraft in two different phases (from both temporal and spatial viewpoints). The reason is that they perform mainly the same general tasks (arrange paper strips, communicate with pilots, supervise the system, solve possible hazardous situations, …), although the specific objects they manage are quite different. For example, both controllers receive requests from pilots. However, while the requests for the ground controller aim to have a path to get to the holding position from the departure's gate (or from the end of the runway to the arrival's gate), the requests directed to the tower controller aim to get the clearance to take-off or to approach the airport.

Then, we started to model an envisioned application which is able to support communication using *data link*, a technology allowing asynchronous exchanges of digital data containing messages coded according to a predefined syntax. The basic idea is to obtain a visual environment that overcomes the limitations of current systems, based completely on voice communication that in some cases can be a bottleneck. At any time only one speaker can broadcast on the frequency, thus, if a pilot is communicating with the controller and some urgent requests arrive from other pilots, they have to wait, and, in some cases, the misunderstanding typical of voice communication in an international environment can occur.

This new solution gives the possibility to provide controllers with real-time representations of the current traffic even when there are bad atmospheric conditions that limit the visibility from the tower. In addition, in the envisioned system we consider the use of enriched flight labels. Their purpose is to replace the paper strips by providing the information concerning a flight directly on the radar screen. They are interactive which means that controllers can select them in order to get additional, more detailed information that is not displayed when they are in the standard mode.

The modelling exercise has considered a large number of tasks: 88 ground controller's tasks, 86 tower controller's tasks, 63 pilot's tasks, and 73 cooperative tasks. Cooperative tasks are tasks that imply actions from two or more users.

3. TASK MODEL-BASED DESIGN

In this section we discuss how it is possible to use information contained in a task model to give support for the user interface design in general terms. In particular, our analysis will focus on two aspects: how to use the temporal operators among tasks to design and implement the dialogue of the corresponding user interface and how to analyse a task and its attributes to identify a suitable presentation technique supporting its performance.

3.1 Analysis of the Operators among Tasks

In this part of the analysis, we focus mainly on the temporal operators. In Figure 2 we give a schematic representation of a task model, where the different types of allocation of tasks and expansion of some subtasks have been neglected (a grey triangle has been put instead of them). For sake of brevity, task names are just represented by letters. The aimed goal is to focus on a portion of a task model little enough not to bore the reader with too many details but sufficient to explain which information the operators give to the designer.

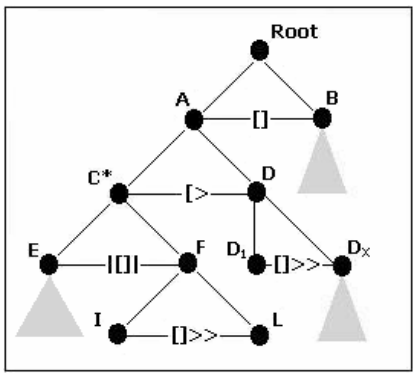

Figure 2: A "simplified" representation of a task model

In fact as you can see from the picture, almost all the temporal operators appear in the selected task model: Enabling ([]>>), Disabling ([>), Interleaving (|[]|), Iteration task*, Choice ([]) so the discussion could be easily generalised and re-applied to other task models.

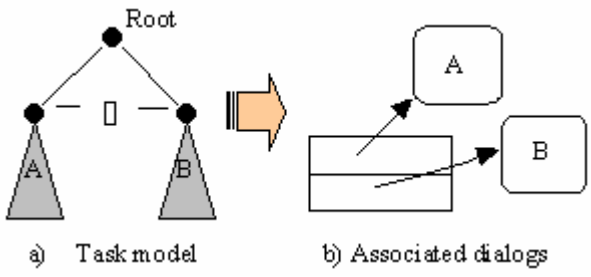

Figure 3: Implementing choice among tasks.

Referring to this "simplified" task model, the presence of the Choice operator "[]" at the highest level means that two possibilities are available. Thus, some suitable interaction technique should be provided for the user to choose from the two clearly display options. Then, the dialogue associated to each branch of the task model can be activated. Using an intuitive graphical language we express the information obtained up to now with the picture in Figure 3.

In the part a) of Figure 3 the task model has been shown highlighting only operators and subtasks at the level that is currently considered (the root level), neglecting for the moment the further specification of each subtask. The part (b) of the picture shows what the task model means in terms of structure of dialogues and presentations. In this case we have that two dialogues (whose structure has been temporarily left unspecified) have to be designed: one is associated to the execution of the subtask A and another one for subtask B. For the moment, the choice of the interaction technique most suitable to *activate* the two different dialogs can be put off (by using two items in a menu —as stylised in the picture— is just the most intuitive example).

Back to the task model and going ahead in its visit, the analysis goes down across the subtask A, whose decomposition is shown in the part (a) of Figure 4.

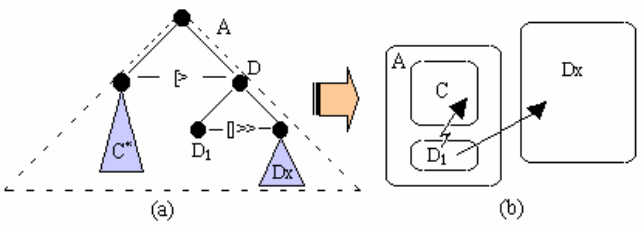

Figure 4: Implementing disabling operator between tasks.

What is the information associated to the task model in Figure 4? The presentation associated to A should be structured in such a way that the activity of the iterative task C could be performed more than one time until the first action of the disabling task D is activated. Note that task D is decomposed into D1 []>> Dx and the first subtask of task D, D1, should be always available to the user during the whole performance of subtask C —in the shape of some interaction technique, e.g. a button as in the picture or something like that. When finally D is started, the presentation should convey to the user the information captured by the task model, which is that, once the D1 subtask has started, the C subtask is no longer available. This effect could be achieved activating another separate dialogue associated to Dx. Down again into the C task, its structure can be viewed as that in the part (a) of the following Figure 5.

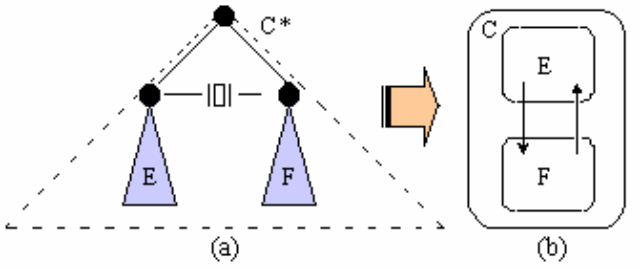

Figure 5: Supporting concurrent communicating tasks.

The meaning associated to the |[]| operator is that the activities are concurrently interconnected each other as they exchange information and in addition, as the parent task can be performed more than one time, it is useful to show the presentations associated to them grouped in the same structure (see part b) in order to highlight how they exchange information each other.

If we summarise the results achieved up to now, we can say that —as far as it concerns the temporal constraints between dialogs associated to subtasks of task A— the presentation should be structured as in Figure 6 (using the same intuitive representation that we used before).

3.2 Analysis of the task

The main purpose of this analysis is to identify the presentation techniques more suitable to perform the task considered. To this end, we need to consider various types of information concerning the task. The type and category identify the type of goal associated with the task and how the performance is allocated. The task model also indicates objects manipulated by tasks with their type and cardinality, and attributes, such as frequency or time-related information.

In particular, the type of task is useful to narrow the space of the interaction and presentation techniques to consider at the implementation level for supporting task performance. For example, spatial tasks (tasks that allow users to provide or manipulate spatial information) should be supported by graphical presentations to improve the immediacy with which convey such information to the user and avoid that the users can perform errors while they handle those data.

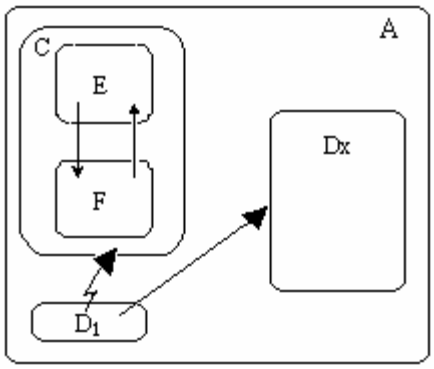

Figure 6: Implementing the example

Another example is that whenever a task manipulates numerical/quantitative data, provide presentations (using graphical attributes) that enhance the performance of typical activities connected with those data (e.g.: comparison). For the ultimate choice about the best presentation designers should consider also the cardinality of data to present.

4. AN EXAMPLE APPLICATION

To clarify our approach we consider an example taken from our case study. We consider the *Build Path* task of the ground controller in the envisioned system, which describes all the activities necessary to answer to a taxi request from a pilot. A taxi request can be presented either by a departing pilot asking for a suitable path to reach the runway from the assigned gate, or by an arriving pilot who has to reach the gate once the aircraft exits the runway. At its highest level the *Build Path* task is decomposed into two main subtasks (see Figure 7):

- *Build path with automatic suggestions*, which describes the activities performed by the controller to build a suitable path exploiting automatic tools (in this case the system really drives the controller to get the best solution showing the set of possible solutions);
- *Build path without automatic suggestions*, when the controller is more self-confident and builds directly the path (in this case the role of the system is mainly to support controllers by helping their decision-making process).

As far as it concerns the subtask *Build path with automatic suggestions*, it is composed of an application task (*Show calculated ordered paths*) concerning the activity of the system to calculate and show the paths ordered by the parameter that is currently the ordering criterion. This activity can be followed by an iterative subtask (*Ordering paths*) that describes all the activities necessary to allow the user to select different parameters and get the set of possible solutions ordered in a different manner. For example, if the controller selects the parameter "Length", the system will show all the possible paths ordered by this parameter. First, the paths that have the minimum length, and then the others in an increasing order (*Show ordered paths*).

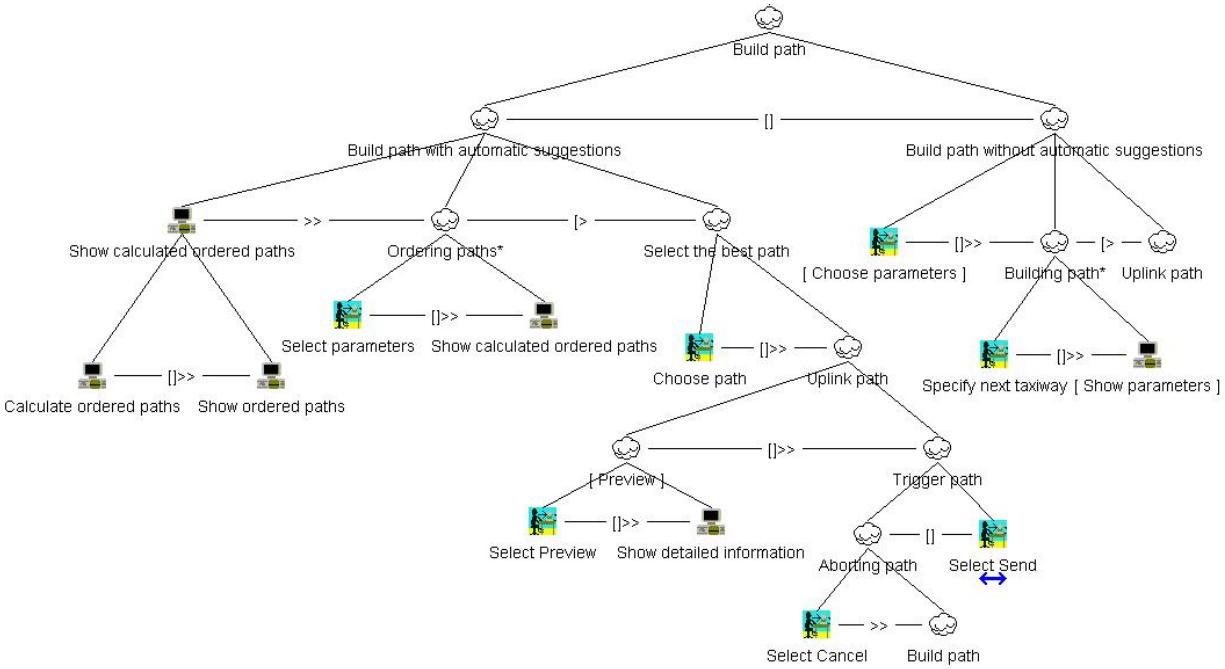

Figure 7: The *Build Path* task model

This process can be repeated multiple times (the task is iterative) because the controller can change mind and select another criterion. However, finally, the controller chooses the "best" path and after having (optionally) activated a preview of all the information about this path (*Preview* task), s/he is able to either send the path or cancel the whole process and restart it. In the task model it is modelled by a recursive instantiation of the *Build path* task. The subtask *Build path without automatic suggestions* allows the controller to build the path specifying directly the taxiways and optionally specifying whether and which parameter s/he is interested to know as s/he gradually builds the path. For example, if the controller has selected the parameter "Length", as s/he gradually selects the various segments of the route, the system updates the overall distance in order to help the controller to decide on the suitability of that path option. The dialogue associated to the *Ordering paths* task is composed of two logical parts, the first one dedicated to the selection of the parameters, and the second one where the set of paths are displayed according to the chosen criterion.

The first dialog is associated to a Selection task (the user has to select item(s) from a predefined set of elements). Thus, the decision about its presentation has to consider this in terms of the possible choices that should be provided, e.g. the cardinality of this set, how they should be provided to the user (single choice, multiple choice). For instance, the possible parameters which the controller could be interested to know are the calculated length of the path, the number of foreseen runway crossings, how much time the travel will take (supposing a standard velocity on the taxiways), and so on.

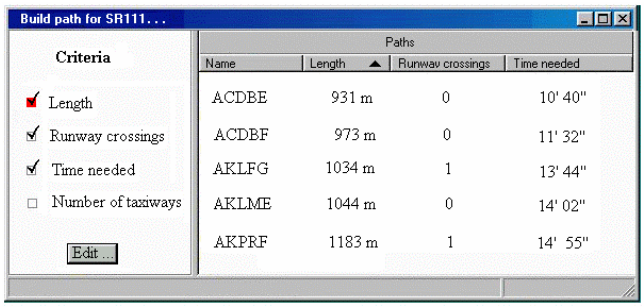

Figure 8: Selection of a path.

The user interface should allow the user to select a list of parameters from a pre-defined set (allowing multiple selections) and mark one of these (single choice) as the sorting criterion. A possible implementation is shown in Figure 8: controllers can mark which parameters are relevant to them (a "√" is associated to each selected parameter) or they can select a different set of parameters ("Edit" button). However, only one parameter can be selected as the ordering criterion, which is highlighted by a different presentation technique in the user interface. There is also the possibility to choose the ordering criterion on the right side of the window, by selecting a specific column, which becomes in such a way the current ordering parameter.

Other considerations have to be done about the type and cardinality of data, being the ultimate choice about the specific presentation that has to be used (especially that exploiting multimedia features) dependent on the integrate consideration of all those aspects. For example, being the path a spatial data, using some graphical technique is a good way to present it. This can be confusing when there are many paths that have to be displayed at the same time. In this case, a good solution is to provide more than one type of presentation, for example an immediate textual path, and the possibility of having a graphical presentation on request. For example selecting a textual representation of a path, then the path is graphically highlighted in a separate window as shown in Figure 9.

As it is possible to understand from the task model, the activity of managing parameters and display accordingly the solutions could be performed more than one time, however, at the end the controller has to decide which is the "best" solution. In order to allow the controllers to perform this activity at their best, the user interface should enhance all the interaction techniques that highlight the comparison of different elements of the same set. In the picture this is modelled by using an ordered list of elements that share the same structure, so it is easy to compare different values referring to the same parameter.

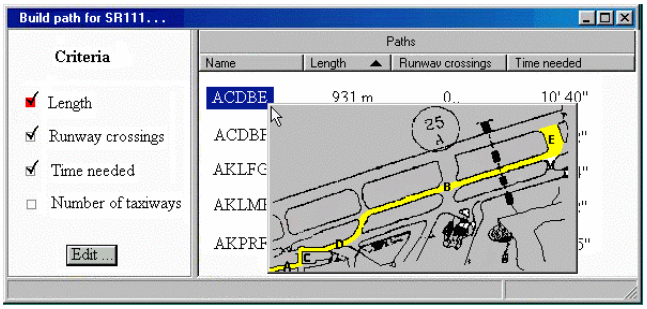

Figure 9: Selection and representation of a path.

When the controller finally selects the path (for example by double-clicking on it as shown in Figure 10), before sending it a preview should be made available to show detailed information (*Show detailed information* task) of the main characteristics of the selected path.

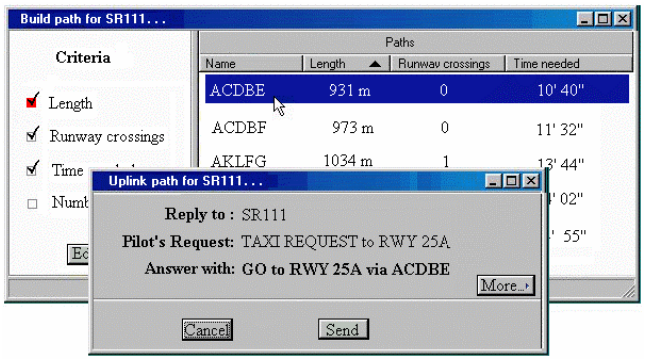

Figure 10: Automatic generation of a command to send.

In Figure 10 we show a possible implementation of these requirements: the controller by double-clicking on a path activates a window where a possible answer for the pilot has been already composed. S/he can decide to send this path to the pilot or not, or to view other information on this path ("More…" button). As this figure is only for the purpose of explanation, we suppose for sake of simplicity that only one predefined format exists for replying to

a taxi request ("GO to RWY <rwy> via <path>"). However, this could easily be extended to enable the controller to select other formats and/or specifying additional options. For instance, once the controller has decided a particular path, s/he could decide to specify other options on when the pilot should start to execute the order.

5. DEVIATIONS AND WARNINGS

We are considering safety-critical applications, thus one important aspect is to support designers to better analyse how the user interface should warn the controller when the system detects some possible hazardous situations. Three main guidelines should be followed:

- *Different levels of warnings/alarms should be distinguished* depending on the different levels of urgency of situations that could arise in the system (a "more" serious hazardous situation requires different presentation techniques with respect to a "less" serious situation).
- *Different media* should be used to convey different information to the controller, exploiting the different nature of the media used;
- *Avoid to overload* the controller too much (e.g. too many different warnings coming from different sources for the same problem) and too often (select the actual hazardous situations in order not to make the controller to get accustomed to see/hear warnings in the system and not to noise them unnecessarily).

Therefore, when possible hazardous situations are identified in the task model (for example, a runway incursion is one of the most serious situations) then the user interface should be designed in appropriate way to exploit its foreseen *multimedia* (audio/visual) capabilities. For example, in case of highly serious situation an *audio* warning is the most appropriate way to be sure to capture the controller's attention as soon as possible. In fact the controller could look at other tools in the system, or out of the window so a visual alarm might not be immediately received. The attributes of the audio signal could be calibrated in such a way to increase its effect depending on the importance.

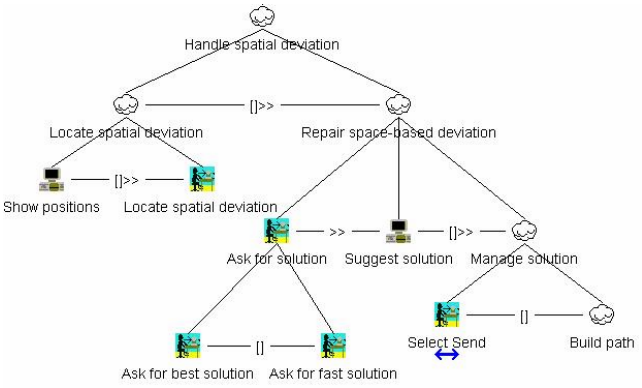

Figure 11: *Handle spatial deviation* task.

For example, the volume of the alarm could be dependent on the level of risk, to be sure that controller is aware of it also in noisy situations. The tone of the alarm could be calibrated in such a way to use high-pitched sounds to reinforce the urgency of the warning, and so on. Thus the audio channel should convey information about how urgent the problem is and how "catastrophic" its possible consequences are. On the other hand, once the controller's attention is captured, the visual channel (exploiting its not-transient nature) should convey additional information about the *cause* of the hazardous situation and, when possible, showing possible *suggestions* about the admissible solutions to the problem itself, although the controller remains the ultimate responsible for the final decision and actions.

As far as it concerns the possible hazardous situations occurring on the taxiways, they can be classified depending on two main criteria: time-based deviations (an aircraft is in the right position but at the wrong time) and space-based deviations (an aircraft is in the wrong position).

With regard to the space-based deviation, the user interface should highlight both the current position of the aircraft and the position where the aircraft was supposed to be (*Show positions* task, see Figure 11), back until where the deviation started to occur. In this way the controller is able to locate the deviation and its extent, focussing his/her attention on the most safety-critical areas (which are those wrongly covered by the aircraft). As in almost the hazardous situations the time is the most safety-critical factor, we suppose that the system is in the best position to calculate the optimum solution to solve the deviation. Thus, a possible action of the controller is to ask the system for the best solution (*Ask for best solution* task in Figure 11): the system shows the best solution and, if it is okay for the controller, s/he has only to send it (*Select Send* task) to the pilot, alternatively s/he should be able to build an alternative path by him/herself. However, when controllers cannot wait too long, they can ask for a fast solution (*Ask for fast solution* task) to be sure to get a solution very rapidly.

As the considered domain is highly safety-critical, the ability of the system to support controllers in handling safely the possible deviations is one of its most relevant concerns. In Figure 11 we have shown the activities that should be undertaken when a general space-based deviation occurs in the system, however, it is useful to distinguish different situations in the set of the space-based deviations. In fact there could be some situations where both the high level of hazard and the short interval of time available to overtake a suitable repairing action force to "accelerate" the beginning of the performance of the repairing action itself. In fact, as you can see from the task model, when a spatial-based deviation has been located (*Locate spatial deviation* task) the controller has to interact with the user interface asking for some solution. Then, s/he has to wait until the system calculates and shows the best solution, and finally s/he can choose if it would be better to send the selected path or to build another solution.

This could be acceptable when controllers have enough time to wait for the time necessary to the system to calculate the "best" solution able to take into account all the specified constraints. However, it can happen that –as the time is a very crucial parameter– sometimes a less "optimal" solution but overtaken earlier can be more helpful in order to solve a hazardous situation. The idea is to distinguish different space-based deviations depending on time constraints. Thus, in case of situations when the controller cannot wait for the "best" solution, the system provides the possibility of activating very soon the first solution

available and some suitable technique should be adopted in order to speed up this activation.

In Figure 12 we have summarised a possible solution on how to handle a spatial deviation. The "abnormal" nature of the situation is highlighted by means of a message that appears above the standard label and with an appropriate colour in order to highlight its urgency and to attract the attention of the controller. The user interface highlights the deviation and allows the controller to get quickly a solution (possibly shown in a graphical way) that the system has automatically calculated to re-integrate safely the aircraft in the traffic flow.

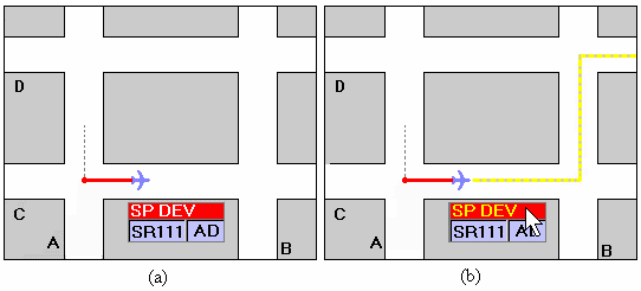

Figure 12: Example of spatial deviation.

6. OTHER TASK-RELATED ASPECTS

In this work we have identified a number of criteria for the design of user interface for safety-critical systems starting from the analysis of the task models. We found that *implementing a user interface dialogue that is consistent with the temporal relationships indicated in the ConcurTaskTrees model* reduces the possibility to introduce errors, which is important especially for tasks with high level of safety-criticality. In addition, the task type and the attributes of the information that is manipulated in the performance of the task are considered.

In this context, it is important to maintain the information always up-to-date and allow users to read/modify different information depending on the activities that have to be performed on a specific object from time to time. In order to reduce the amount of information to provide permanently to the user, it is important to *define levels of priority amongst data*. This allows designers to limit the permanently displayed data only to those that are necessary to get the overall picture of the current and future situation and gives users the possibility to get additional information only after an explicit action of the user.

We have seen how to provide *different levels of warnings/alarms* for each deviation that could occur depending on the impact on the safety of the system. The features of each medium can be used to convey different information to the user about hazardous states (e.g. audio media to attract attention, visual media to suggest solutions). This allows designers to avoid overloading users with too many alarms, and making them accustomed to warnings.

If it is possible to identify different types of users that perform different activities manipulating the same type of data, then it is important to design *different user interfaces for each of them* in order not to provide them with meaningless information and useless interactions. This implied that we had to design different labels for the two controllers.

We have used additional task-related criteria in the design of our case study. One aspect considered is when there are several tools that offer different, partial "views" of the same object. Then, we have to provide users with an automatic link that allows them to get an *immediate correlation among the different views* giving the most complete picture of the object whatever tool is considered. While each view is more oriented to support a specific task, it is useful that they are able to support also those tasks that are primarily performed by the other views. We applied this in defining the relationships among different tools for the tower controller in the new environment. Such tools are the set of enriched flight labels, the list of data link commands received from pilots and a departure manager that helps controllers to schedule the departing flights.

Figure 13: An example of a data link clearance.

Finally, another important aspect is to *change the task allocation from the human to the machine* for tasks (especially routine tasks) that force users to distract their attention from the most safety-critical activities. For example, while in current environments the ground system containing flight information is manually updated whenever a flight parameter changes, with data link technology it is possible to automatically detect such changes and update the ground system.

The method applied implied the development of task models that required some effort and time. However, this allowed a more rigorous understanding of the design implications and a more extended analysis of possible safety-critical issues that is particular important given the nature of the application considered and would have been more difficult to achieve with cheaper techniques. The design method was particularly useful to make the main decisions concerning the structure of the dialogue and presentation of the user interface. Some low-level details concerning the user interface were designed following general design rules as well.

The method has been used in the development of a real prototype, MIDAS (Mefisto Interface Development for Aerodrome Systems) which has been implemented by Alenia Marconi Systems.

Figure 13 shows an example of the user interface of the current prototype. We can see how the system provides information useful to check if the aircraft is on the right path (the path sent by the controller and stored in the system).

7. CONCLUSIONS

In this paper we have discussed the use of task models to support visual design of interfaces for interactive safety-critical applications. We have seen how this approach can give useful suggestions to designers while they can still tailor them for the specific case study considered.

Future work is planned to further refine such criteria to support more extensively the design of low-level user interface aspects. We also plan to extend our method to support the usability evaluation phase taking into account the specific features of the type of application we are considering (safety-critical applications).

8. REFERENCES

[1] F.Bodart, A.Hennerbert, J.Leheureux, J.Vanderdonckt, "A Model-based approach to Presentation: A Continuum from Task Analysis to Prototype", in F. Paterno' (ed.) *Interactive Systems: Design, Specification, Verification*, pp. 3-14 Springer Verlag, 1994.

[2] Booch, G., Rumbaugh, J., Jacobson, I., *Unified Modeling Language Reference Manual*, Addison Wesley, 1999.

[3] Chatty, S., Lecoanet, P. (1996) Pen Computing for Air Traffic Control, in *Proceedings of CHI'96*, April 13-18, 1996 Vancouver, British Columbia, Canada.

[4] Foley, J., Sukaviriya, N., "History, Results, and Bibliography of the User Interface Design Environment (UIDE), an Early Model-based System for User Interface Design and Development", in F. Paterno' (ed.) *Interactive Systems: Design, Specification, Verification*, pp. 3-14 Springer Verlag, 1994.

[5] Mackay, W.E., Fayard, A.L., Frobert, L., Médini, L. (1998) Reinventing the familiar: exploring an augmented reality design space for air traffic control, in Proceedings of CHI'98, April 18-23, 1998, Los Angeles, CA USA.

[6] Paternò F., Model-Based Design and Evaluation of Interactive Applications, ISBN 1-85233-155-0, Springer Verlag, 1999.

[7] Puerta A., Cheng E., Tunhow O., Min J., MOBILE: User-Centred Interface Building, *Proceedings ACM CHI'99*, pag.426-433, ACM Press, 1999.

[8] Szekely, P., Luo, P., Neches, R., "Facilitating the Exploration of Design Alternative: The HUMANOID Model of User Interface Design", *Proceedings CHI'92*, pp. 507-515, ACM Press, 1992.