Formal Tasks and Systems Models as a Tool for Specifying and Assessing Automation Designs

Célia Martinie, Philippe Palanque, Eric Barboni, Marco Winckler
University Paul Sabatier, IRIT
118, route de Narbonne
31062 Toulouse Cedex 9
{martinie, palanque, winckler}@irit.fr

Martina Ragosta, Alberto Pasquini, Paola Lanzi
DeepBlue Srl
Piazza Buenos Aires 20, 00198 Roma - Italy
{martina.ragosta, alberto.pasquini, paola.lanzi}@dblue.it

ABSTRACT
Designing systems in such a way that as much functions as possible are automated has been the driving direction of research and engineering both in aviation and in computer science for many years. In the 80′s many studies (e.g. [1] and [8] related to the notion of mode confusion) have demonstrated that fully automated systems are out of the grasp of current technologies and that additionally migrating functions [2] from the operator to the system might have disastrous impact on both safety and usability of operations. Allocating functions to an operator or automating them, raises issues that require a complete understanding of both operations and the behavior of the interactive system. This paper proposes a contribution for reasoning about automation designs using a model-based approach exploiting both task models and system models. Tasks models are meant to describe the tasks and actions to be performed by the operator while system models represent the entire behaviour of the interactive system. Tasks models and systems models thus represent two different views of the same world: one or several users interacting with a computing system in order to achieve their goals. In previous work we have demonstrated how these two views can be integrated at the model level and additionally at the tool level [7]. In this paper we present how such representations can support the assessment alternative design options for automation.

Keywords
Interactive critical systems design, formal models, levels of automation.

1 INTRODUCTION
Nowadays, users of safety critical systems are facing more and more sources of information competing for attention which might affects their abilities to complete their tasks.

Automation (i.e. delegation of user’s tasks to the system) can reduce tasks’ complexity and time consumption allowing operators to focus on other tasks. However, too much (or inadequate) automation can lead to complacency, loss of situational awareness, or skill degradation, whereas not enough automation can lead to an unmanageable, unsafe or problematic workload [10].

Due to the fact that system automation can have a huge impact on human performance, there is a need for methods and tools making it possible to assess the impact of automation levels at design time. In the Human-Computer Interaction (HCI) field, there is a consensus (both in industrial and academic communities) on the importance and usefulness of providing designers with complete and unambiguous descriptions of both users’ tasks and system. One of the ways of reaching this goal is to use models in the design and development process of interactive systems. Task models, such as CTT [9] and HAMSTERS [1] have proved useful in expressing in an exhaustive manner the goals of the users and the activities they are expected to carry out in order to reach these goals. System models describe important aspects of the user interface such as the set of states the system can be in, the set of actions the system is able to perform, the set of events the system is able to react to and the state changes that occur when such events or actions are performed. More generally, system models aims at helping designers to build the application (by describing the in details its behavior) before it is implemented. These two models have to be embedded in the development process of interactive systems in a complementary way as they correspond to different views on the same world (one being centered on user behavior and the other being centered on system’s behavior).

In this paper we describe how the synergistic use of these two representations can be fruitfully used for the design and the assessment of various design of automation. Section 2 provides an overview of task models, system models and how they can be related to each other in order to integrate the operator view with the system view. Section 3 presents a case study demonstrating how these models can be used to support the description and the analysis of automation designs. A discussion on the advantages and limitations of
the approach is presented in section 4 together with a description of how the work presented here is related to previous work in the field. Lastly, due to the fact that this paper is only an abstract, it also concludes the paper and proposes an outline of future work.

2 MODELS SUPPORTING THE DESCRIPTION OF AUTOMATION DESIGN

Models represent an abstract view of what they are aimed at describing. Such abstraction makes it possible for the designer to have a representation of the system avoiding to deal too early in design process with too much details.

This section presents a very short description of tasks models and interactive systems models. While it focuses on two formalisms (HAMSTERS and ICOs) the concepts introduced here would hold for many others formalisms. However, both HAMSTERS and ICOs exhibit specificities that allow, for instance, going from the abstract model to the implementation of the interactive application. We believe this is a critical characteristic as it avoids possible discrepancies between the abstract representation and the concrete application.

2.1 Task Models

Task models are aimed at supporting the design and evaluation of user centered applications and appliances [4]. In our approach (of which a diagrammatic representation is proposed in Figure 1), the users’ activities and goals corresponding to the missions they have to perform, are detailed in task models using HAMSTERS\(^1\) notation and tool [1]. This notation and tool enables structuring users’ goals and sub-goals into a hierarchical tasks tree in which qualitative temporal relationship amongst tasks are described by operators. Goals or sub-goals are modeled using the type of task called “abstract”. An abstract task can be refined in 3 types of tasks: “user task”, “system tasks” and “interactive tasks”. A “user task” can be refined in the following sub-types: “perceptive task”, “cognitive task” and “motor task”. An interactive task can be refined in the following sub-types: “input task”, “output task”. Figure 3 shows an example of such models. The element at the root of the tree (called “ManageWXRApplication”) corresponds to a goal to be reached and is thus of “abstract task” type. In order to reach this goal the operator has to perform many actions of various types that are described in the lower part of the tree.

2.2 System models

ICO [6] is a formal description technique dedicated to the modeling of interactive applications. This formalism makes it possible the entire interactive application including both the behavioral aspects (states and state changes) and the interaction aspects (events triggered on the user interface and graphical rendering). It is based on objects Petri nets. An example of the behavioral description in ICOs is given in

\(^1\) http://www.irit.fr/recherches/ICS/softwares/hamsters/

Figure 4 and Figure 6.

PetShop\(^2\), is the CASE tool associated CASE to the ICO formalism. It allows editing models and their execution. The models of

Figure 4 and Figure 6 have been edited using PetShop. In conformance with Petri nets, ellipses correspond to places (and support the description of the states the system can be in) while rectangles are called transitions and correspond to the action the system can perform. Transitions are connected to places representing the fact that some actions (represented graphically by transitions) can only be performed if the system is in a given state.

2.3 Articulation between models

Our approach, summarized in

Figure 1, is based on the synergistic integration of the tasks and system models. The foundations of this integration have been proposed in [7], while the effective integration between HAMSTERS and ICOS has been developed in [1]. The suite of notations and tools presented in the two previous paragraphs, Petshop and HAMSTERS, allows editing the correspondences between task models and system models, and then to identify at runtime, which steps of the execution on the task model and on the system model is currently being performed.

\(^2\) http://www.irit.fr/recherches/ICS/softwares/petshop/
If some functions are “migrated” to the system model, then the task model has to be mended in order to take into account this migration.

The resulting tasks and systems models have to be checked for compatibility (represented by the box “correspondence” at the top of the diagram in Figure 1). This guarantees the consistency between the system and the user, i.e. that there are no combination between users’ tasks and the system’s actions.

3 SYNERGISTIC USE OF TASK AND SYSTEM MODELS: A CASE STUDY ON WXR

To illustrate the approach presented above, we will apply it to an example from the domain of interactive cockpits. We will use an application currently deployed in many cockpits of commercial aircrafts called WXR (Weather Radar System).

3.1 Informal description

Figure 2. Screenshot of the WXR application

Figure 2 presents a screenshot of the WXR application. This application provides two functionalities to the crew members. The first one, on which we will focus, is dedicated to the mode selection of weather radar. The operation of changing from one mode to another one can be performed in the upper part of the window. The second functionality, available in the lower part of the window, is dedicated to the adjustment of the weather radar orientation.

The crew members have to be aware of the running status of the application, in order to ensure that the weather radar can be set up correctly. Some tasks such as the testing of the weather radar are rather repetitive and of limited interest with respect to the piloting activity. In this section dedicated to the case study we will describe how the testing of the WXR application could be automated and how both HAMSTERS and ICOs can support the precise and unambiguous description of such migration of function from the operator to an autonomous part of the system.

3.2 Designing a first iteration of the WXR application

Figure 3 presents an excerpt of the task model describing the pilot’s activities for managing the WXR application (due to space constraints, the manage tilt angle sub-parts are folded, as showed by symbol). As explained above this task model is hierarchical and the temporal relationships are represented by means of operators i.e. symbols such as >> for a sequence between two tasks. From the left sub tree “test the WXR application”, we see that the crew can periodically decide (“decide application needs to be tested” “cognitive task” type) to switch from current application mode to test mode. This action on the task model corresponds to the TST radio button of the interactive application presented in Figure 2. Once “switch to TST” “input interaction” task has been performed a graphical notification from the system informs them about the status of the application. It can be either “notice that WXR is OK” or “notice that WXR is KO” both tasks being of “output interactive” type. If the status is incorrect (the test has
failed) they might decide to reset the WXR application.

Figure 4 presents the ICOs model corresponding to the behavior of the interactive part of the WXR application. This application allows crew members to modify the current mode of the application. A click event on a radio button (OFF, STDBY, TST, WXON or WXA) triggers the corresponding transition (off_T1, stdby_T1, tst_T1, wxon_T1 or wxa_T1) in the model. As defined by the arcs, once triggered, a transition takes the token from place MODE_SELECTION, changes its value and puts it back in the place. When the token is deposited in the place, the rendering function changes the application visual appearance according to the token value. In this case, a black disc appears included in the grey disc of the selected radio button (see “Off” radio button in Figure 2).

![Figure 4. System model of the mode selection part of the WXR application](image)

As stated above, when both the task model and the system model have been edited, a correspondence is defined by, for instance, connecting “interactive input” tasks with system model transitions and “interactive output” tasks with system rendering. This correspondence enables a first compatibility check between the interactive functionalities that the system is providing and the tasks that the users have to perform.

3.3 First analysis for automation design: task migration

When analyzing the task model in Figure 3, we can see that the three main activities for the crew members are: “test WXR application”, “manage modes” and “manage tilt angle”. The first operation is mainly relying on information acquisition and action implementation function types of the Parasuraman four-stage model of human information processing [10]. Furthermore, as this operation is quite repetitive and has to be handled periodically, in might occur concurrently with the other two operations and thus, depending on their workload, the crew members might forget to perform the test. This functionality is thus a good candidate for migration and we propose automate it partly. Indeed, in order to keep the members aware of the status of the application, analysis and decisional sub-tasks are not automated.

3.4 Second iteration of the WXR application

Figure 5 represents the task model corresponding to the tasks associated with the partly automated version of the WXR application. In that case the crew members don’t have to handle the application testing which is now performed automatically by the system. This is represented in the task model by the added “system” task called “WXR application auto testing”. However, the crew still has to check that the auto testing has been completed successfully (as in the manual testing case).

Figure 6 represents the new version of the system model. A new place has been added, “AUTO_TESTING”, as well as a new transition “autoTest_T1”. The time parameter [2000] of that transition models the fact this action will be performed every 20 seconds (and is not related to crew events on the user interface). After this check (once the

![Figure 5. Task model of the manage WXR application activity when application testing has been automated](image)
token comes back to place “MODE_SELECTION”) the rendering function of the model updates the visual appearance of the application depending on the token value. For example, if the value of “new_ms” token is negative (meaning that the test failed) the rendering function will display every radio button and associated label in red, so that the crew members notice it which is modeled by the “notice WXR is KO” interactive output task in Figure 5.

Figure 6. System model of the mode selection part of the WXR application with the automated testing

As for the previous examples, when the models have been built they are connected to assess their compatibility. The results of the qualitative analysis now fulfill Parasuraman criteria and the application could be carried out for usability and operation testing (as represented at the bottom of the process described in Figure 1).

4 DISCUSSION AND RELATED WORK

A lot of work has been carried out in the past in the area of Automation. Parasuraman and al. [10] have proposed a classification of level of automation, a simplified model of human information processing and evaluation criteria as a framework for automation design. In [11], Proud and al. proposed the LOA (Level Of Autonomy) Assessment Tool (based on a LOA Assessment Scale) which outputs analytical summaries of the appropriate Level of Autonomy for particular functions of an Autonomous Flight Management system. Cummings and al. [3] identified a refinement mechanism for the decision making step, to help in deciding which one of the human or of the system should perform a given decision task. Lastly, Johansson and al. [5] developed a simulation tool to analyze the effect of the level of automation and emphasize the importance of a simulation framework to have a feedback on design choices before deploying the system.

Our approach supports this philosophy as 1) it enables to analyze and test the conformance of the actions that have to be distributed between the user and the system and 2) it enables to perform simulations of the designed application with real users.

The case study has presented both the task and system models of two design iterations of an interactive cockpit application. These models have been analyzed in order to identify potential candidates for automation.

The final paper will extend this abstract by providing more information about the analysis principle especially providing criteria and factors of success or failure of automation designs.

5 REFERENCES