HAL open science

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

Célia Martinie, Philippe Palanque, Eric Barboni, Marco Winckler, Martina Ragosta, Alberto Pasquini, Paola Lanzi

To cite this version:

Célia Martinie, Philippe Palanque, Eric Barboni, Marco Winckler, Martina Ragosta, et al.. Formal Tasks and Systems Models as a Tool for Specifying and Assessing Automation Designs. International Conference on Application and Theory of Automation in Command and Control Systems, May 2011, Barcelone, Spain. pp.50-59. hal-03647159

HAL Id: hal-03647159

https://hal.science/hal-03647159v1

Submitted on 20 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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 interactive computing 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. [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 safety and usability and operationality of systems. Allocating functions to an operator or automating them, raises issues that require a complete understanding of both operations to be carried out by the operator 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 goals, 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 of alternative design options for automation.

Categories and Subject Descriptors
I.6.4 [**Model Validation and Analysis**]. H.5.2 [**User Interfaces**]: *Evaluation/methodology*.

General Terms
Human Factors, Performance.

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

Copyright © 2011 IRIT PRESS. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Re-publication of material on this page requires permission by the copyright owners.

ATACCS'2011, 26-27 May 2011, Barcelona, Spain.

1 INTRODUCTION
Nowadays, operators of safety critical systems are facing more and more sources of information competing for attention which might affect 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. Indeed assessing automation designs later in the development process might result in requirements for changes too late for making it possible to integrate them. In the field of Human-Computer Interaction (HCI), there is a consensus 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. Models make it possible to represent in an abstract and high-level way information and are (most of the time) associated with tools that allow reasoning on the models.

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. Such detailed description covers the behavioral aspects of the system but also how this behavior is related to the user interface both in terms of output (how states and state changes are represented to the users) and input (how users can trigger system actions while interacting with the input devices).

These two models have to be embedded in the development process of interactive systems in a complementary way as they correspond to two different views on the same world (one being centered on

operator's 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 several designs of interactive systems featuring autonomous behaviors. The contribution is done at two different levels: first at the notation and tool levels by showing how the existing notations ICO and HAMSTER can be extended to integrate descriptions of autonomous behaviors. Second a development process explicating where automation designs take place and how the notations (and their associated tools) are used within this development process.

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.

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.

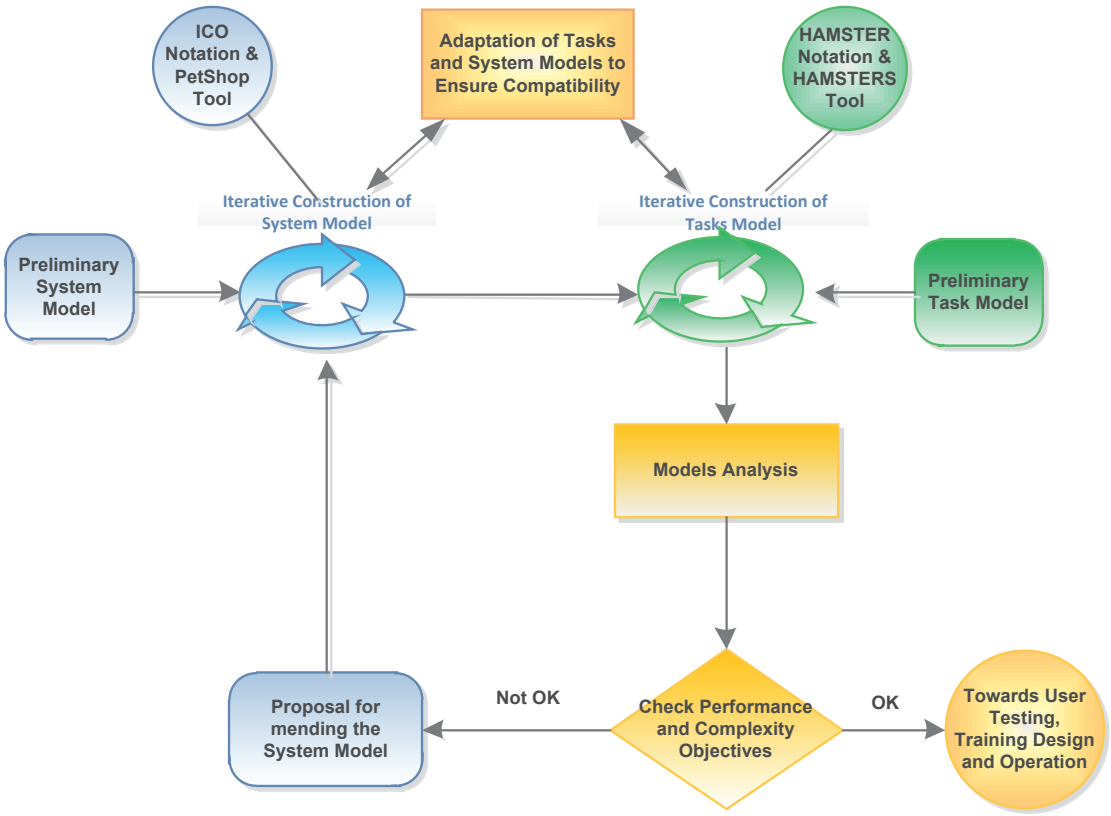


Figure 1. The iterative model-based design life cycle using both tasks and system models

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.

2.1.1 Modeling requirements for describing tasks

Within a model-based UI development methodology, the creation of the task model is a commonly agreed-upon starting point [21]. Tasks analysis is a central element of user centred design approaches. For this reason a lot of

[1] http://www.irit.fr/recherches/ICS/softwares/hamsters/

work has been devoted to it and to its integration in the development process of interactive systems. Analyzing the tasks undertaken by current operators and the tasks to be accomplished by operators of a future system is necessary to ensure client requirements are met and satisfy the end users.

A task model is a representation of user tasks often involving some form of interaction with a system influenced by its contextual environment. We use the word "influenced" (as opposed to "driven" by the environment) to highlight our thoughts on user's having an underlying goal and hence plan in their mind before attempting to perform a task. This contrasts Suchman's [21] theory of situated action. This theory analyses user behavior through emergent actions of users during a particular activity. Situation action dismisses the role of predetermined intentions and goals of a user as part of the analysis. There is no intentionality in situation action since what happens is always developing ad-hoc out of the current situation. While Suchman's view may be true for gaming and leisure activities it is clear that in the application domain considered in this work, training goals, performance and task efficiency are critical to the correct and safe operation of the system.

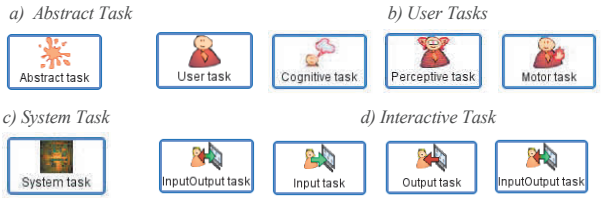
Notations for describing tasks models should be powerful enough to encompass all the information that is necessary for describing user's activities while interacting with computer systems. This includes:

- Structuring mechanisms for representing activities in a hierarchical way and for making it possible to structure large task models;
- The description of artefacts used to perform a task should be close to the representation of objects manipulated by the system;
- User tasks should include elements of the behaviour expected from the system; e.g. user providing an input to the system, requesting a feedback or any kind of system output, or both actions at the same time.
- Task models should be able to express both qualitative temporal relationships (e.g. task ordering such as concurrency, sequence, interleaving, …) and quantitative temporal relationships (e.g. amount of time required to perform a task). These relationships are needed to describe time constraints applied during system execution;
- It must be possible to describe tasks models as unities that cooperate rather than monolithic models. This aspect would support a better mapping between tasks and different system's modules.

2.1.2 *Overview of the HAMSTER formalism*

HAMSTERS is an acronym that stands for Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems. It is inspired from existing notations, in particular from Concur Task Trees (CTT) [9] and has thus been intended to remain compatible (at the users level) with it. Indeed both can be considered as hierarchical and graphical models representation relationship between tasks by means of operators (see Table II). However, HAMSTERS involves extensions such as conditions associated to task executions, data flow across task models etc. extending its expression power beyond the one of CTT. Additionally, it is publicly available, featuring a task simulator and providing a dedicated API for observing editing and simulation events.

Table 1. Tasks Types in Hamster notation



As summarized in Table 1, the elements of task models in HAMSTERS include:

- Abstract task: a task that involves sub tasks of any types.
- System function: a function performed only by the system.
- User task: a generic task describing a user activity. It can be specialized (from left to right on Table I) as Motor task (e.g. pressing a button), Cognitive task (e.g. comparing value, remembering information), or Perceptive task (e.g. reading some information).
- Interactive task: a task describing an interaction between the user and the system. It can be refined (from left to right on Table I) into Input task when the users provide input to the system, Output task when the system provides an output to the user and InputOutput task (both but in an atomic way).

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 5 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.

As for CTT, each task in HAMSTERS can be iterative, optional or both (as graphically shown in Figure 2).

Figure 2. Icons of Optional, Iterative and both iterative and optional tasks

More precisely iterative refers to a task that can be executed one or several times but can be interrupted or suspended by another task. An optional task is a task that does not necessarily needs to be executed. During the simulation, an optional task will be proposed with the following task(s) to be executed. Again, as in CTT temporal relationship between tasks is represented by means of operators as described by Table 2.

Table 2. Illustration of the Operator Types in Hamsters

Operator/ type	Symbol	Description
Enable	>>	ENABLE operator allows its tasks and/or task group and/or operator groups to execute one after the other, from left to right.
Concurrent	|||	CONCURRENT operator allows tasks and/or tasks belonging to task groups and/or operator groups to execute "at the same time" in any order.
Choice	[]	CHOICE operator allows the user to select the first available task to execute among each available sub-branch. When a task is executed, HAMSTERS disables all the other branches that don't contain the executed task.
Disable	[>	DISABLE operator shall deactivate the execution of the first branch when a task is executed on the second branch. DISABLE operator shall have 2 and only 2 branches.
Suspend-resume	|>	SUSPEND-RESUME operator suspends the execution of the first task or branch when task is executed on the second branch.
Order Independent	|=|	ORDER INDEPENDENT operator allows its tasks and/or task groups and/or operator groups to execute one after another, in any order.

In HAMSTERS, the notion of object represents the elements of the world manipulated by tasks. HAMSTERS offers constructs for representing the information flow between tasks. One example of such information flow can be seen in Figure 3 with the symbol "TC list" representing a list of information flowing from one task to another one.

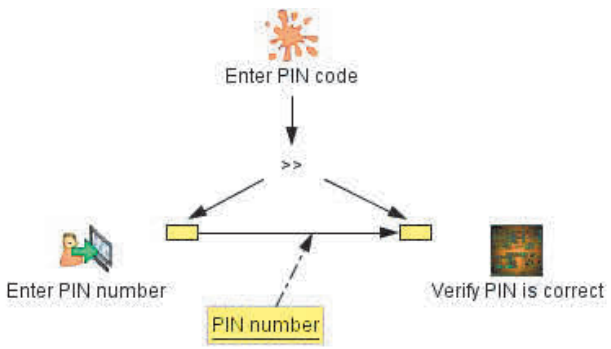


Figure 3. Input (right-hand side of a task) and output (left-hand side of a task) flow in HAMSTERS

Extension for handling large task models

Task modeling activities become cumbersome and hard to manage when performed on large, real-life systems. However, one of the main goals of task models is to provide designers with a structured and complete description of the users tasks especially when these user tasks are numerous and/or complex. In [22], we proposed structuring mechanisms to support the effective exploitation of task models for large scale application.

Extension for dealing with automation

HAMSTERS notation enables to model in a structured manner human activities to accomplish a goal. To accomplish the modeling, a task type has to be associated to each stage of the Parasuraman model of human information processing [10] as well as other models such as the action theory [24] and the human processor model [23].

In case of close loop between perception and action (in case of low level interaction with a graphical widget for instance) such human activity is represented by InputOutput tasks (see last icon in Table 1). In the previous version of HAMSTERS, "Perception/working memory" and "Decision making" system functions can only be modeled as cognitive tasks. It was thus not possible to describe in detail users' tasks if automation has to be considered. In order to describe more precisely these two aspects, we have introduced two new sub-types of cognitive task (Figure 2):

- Perception/working memory is represented with a cognitive analysis task (left-hand side of Figure 4).
- Decision making is represented with a cognitive decision task (right-hand side of Figure 4).

Figure 4. Illustration of Cognitive analysis and decision task types

More detailed information on how automation can be represented and integrated within iterative processes for analyzing and designing interactive satellite control room applications can be found in [25].

2.1.3 Tool support with HAMSTERS

Beyond the notation, HAMSTERS is a tool making it possible for designers to edit the tasks models but also to run them. This simulation functionality is critical as it makes it possible to see directly the behavior of the tasks model. These runs correspond to scenarios extracted from the task model. These scenarios can be stored and reused for further purposes such as non-regression testing when task models are modified or included in the training program of the operators.

2.2 System models

In our approach, system modeling is done using ICO [6] which is a formal description technique dedicated to the

modeling of interactive applications. This formalism makes it possible to describe the entire interactive application including both behavioral aspects (states and state changes) and interaction aspects (events triggered on the user interface and graphical rendering).

Interactive Applications design is bringing the user's perspective in system-centered software engineering. This section highlights the reasons why we use a Petri nets-based formalism to model Interactive Applications and details the key elements of the formal notation we are using.

2.2.1 *Modeling requirements for Interactive Applications*

Formal description techniques have proven their value in several domains and provide a unique support to understand, design and develop systems and check their properties. Nevertheless, as detailed in [13], Interactive Applications feature specificities that have to be taken into account by formal description techniques. Some of these constraints come from the fact that requiring additional modeling conditions to ensure their usability and reliability:

- As users will be interacting with the Interactive Application in an asynchronous and non-predictable way, we need a representation that allows the modeling of **concurrent** input and output users' actions. This is even more critical for interactive cockpits as both the pilot and the first officer have a KCCU at their disposal. The CDS behavior must be fully multimodal allowing, for instance, the synergistic handling of two input device as well as the fusion of information they produce. A detailed presentation of such requirements is available at [14].
- The notation has to be capable of dealing with the **event-driven** architecture of interactive systems as interactions between users and Interactive Applications will take place through events produce through users' actions on the input devices.
- The notation has to be able to describe in a complete way all the **states** of the Interactive Application and how the various events are leading to state changes.
- The notation has to be capable of representing and manipulating in an integrated way **data structure** and control structure of the application as these applications manipulate a large quantity of information and as this information influences their behavior.

2.2.2 *Overview of the ICO formalism - previous use and grounding of ICOs*

The Interactive Cooperative Objects formalism is compliant with the requirements introduced in previous section and can be decomposed in 4 elements.

First, it is Petri Net based, and then suitable to specify the behaviour of event driven-interactive systems and concurrent human computer interactions and to describe the inner states of the Interactive Application. The interested reader is encouraged to have a look at [15] for a complete description of this point. The formalism also supports two arc extensions [16]: test arcs and generalized inhibitor arcs.

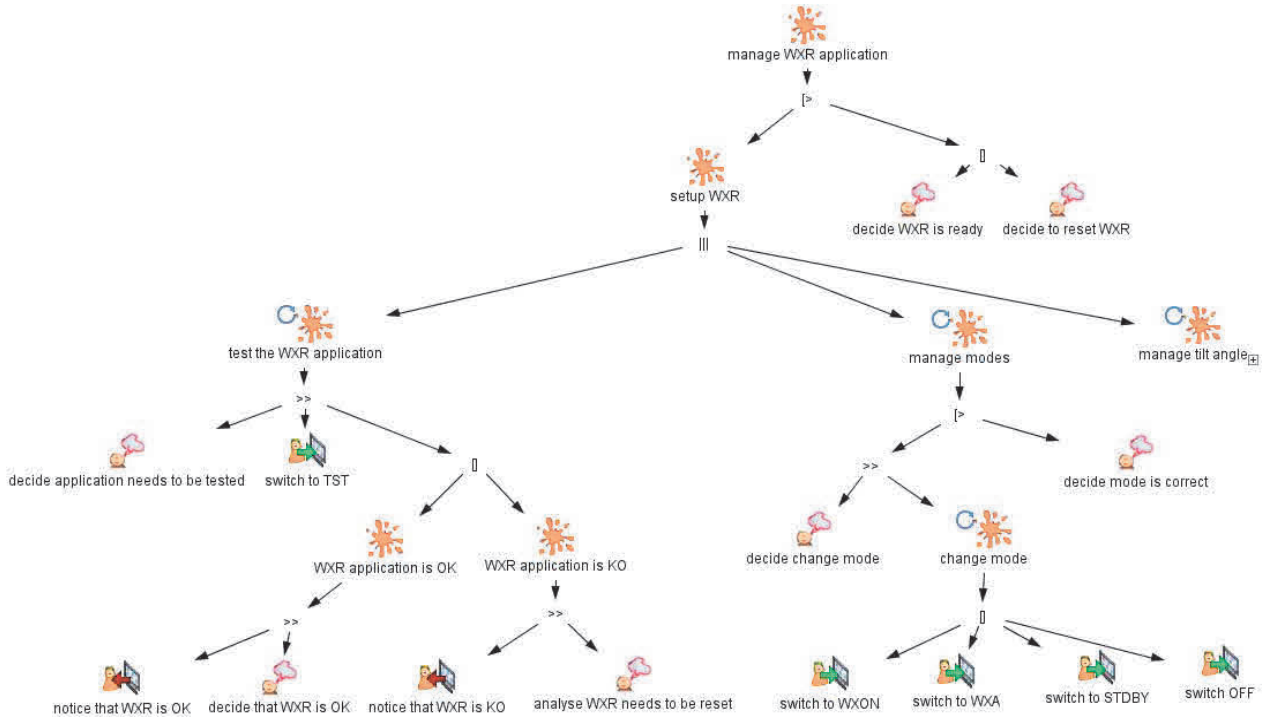


Figure 5. Task model of the manage WXR application activity

Second, it is built upon the OO Petri nets paradigm [17], which enables the handling of more complex data structure (typed places and tokens, transitions with actions and preconditions, variable names on arcs).

Third, software interface description and communication capabilities are added to OO Petri nets. This new type of

OO Petri net is called Cooperative Object [15] and allows objects of this type to react to external events according to their inner state and to produce events. They are also able to offer services that can be called by of COs. Such communication follows the client-server protocol pattern introduced in [18] and is described using Service Input Port and Service Output Port formalism.

Lastly, the ICO formalism [6] defines an object as the set of 4 elements: a Cooperative Object, a presentation part, an activation function and a rendering function.

The presentation part defines the external appearance of the object; it ranges from a set of windows to a single widget. The activation function associates a given input event (from the user action on the input device) to the corresponding Cooperative Object service. The rendering function associates a change in the inner state of a Cooperative Object to an output to the user via the graphical interface.

This formalism has been used in various application domain and for describing various types on interaction techniques and user interfaces. It presents the following additional advantages:

- The specification encompasses both "input" aspects of the interaction (i.e. how user actions impact the inner state of the application, and which actions are enabled at any given time) and "output" aspects (i.e. according to which state change the application displays information relevant to the user). Input is typically event-based while output is state-based.
- The specification is fully executable and modifiable at runtime, which gives the possibility to prototype and test an application before it is fully implemented [19].

An example of the description of the behavioral part of ICOs is given in Figure 8 and Figure 9.

2.2.3 *Tool support with PetShop*

PetShop[2], is the CASE tool associated CASE to the ICO formalism. It allows editing models and their execution. The models of Figure 8 and Figure 9 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. A precise description of the structure and functioning of PetShop can be found in [20].

2.3 Articulation between models

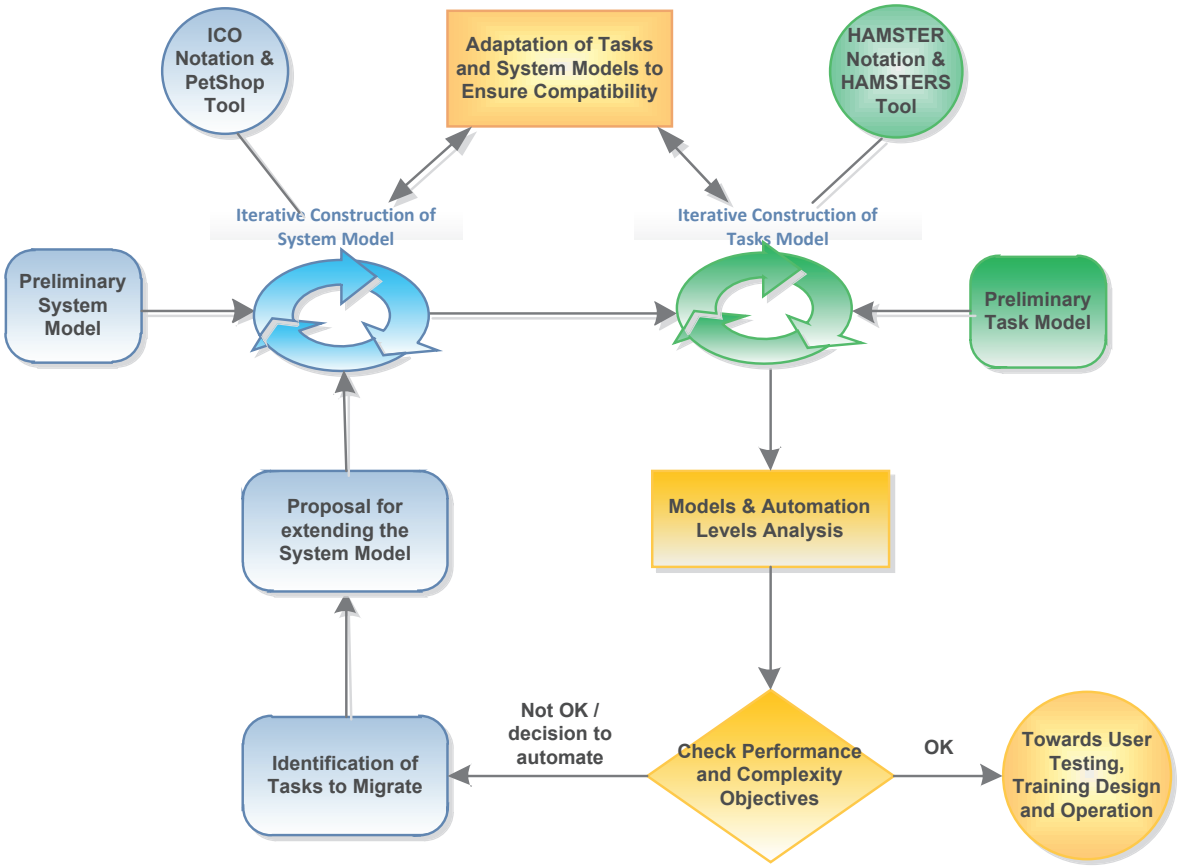


Figure 6. The iterative model-based design life cycle exhibiting automation design activities

Our approach, summarized in Figure 1, is based on the synergistic integration of the tasks and system models. While system models and tasks models might be developed independently, the process exhibits the necessity to ensure conformance and compatibility for these two views. For instance each interactive task in the task model should correspond to an interactive object on

[2] http://www.irit.fr/recherches/ICS/softwares/petshop/

the user interface and thus in the system model. Beyond that, sequences of action in the task model should be accepted by the behavioral description of the system model.

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.

In Figure 1, the left-hand part of the diagram corresponds to the system part while the right-hand side corresponds to the task part. The modeling process can either start with a preliminary system model or with a preliminary task model. In such a case, the task model complexity is assessed (analysis box of the diagram). If the complexity is too high, then the system has to be improved by, for instance, including more functions (that might have been previously attributed to the operator i.e. represented in the task model). 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 "Adaptation of Tasks and System Model to ensure Compatibility" at the top of the diagram in Figure 1). This guarantees the consistency between the actions and sequence of actions offered by the user interface of the system and the user's goals and activities.

When both the systems models and the tasks models have been produced and their compatibility has been assessed the tasks models are analyzed in terms of complexity and performance. Indeed, the system that has been produced might cover all the tasks of the users but these tasks might remain too cumbersome and error prone. If the analysis exhibits such results then the system models have to be modified. Such modifications will have to be transmitted to the tasks models as the tasks are heavily dependent (at least at the lower level of the task tree) on the system they are meant to be executed on.

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 7 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.

Figure 7. Screenshot of the WXR application

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 5 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 8 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 7).

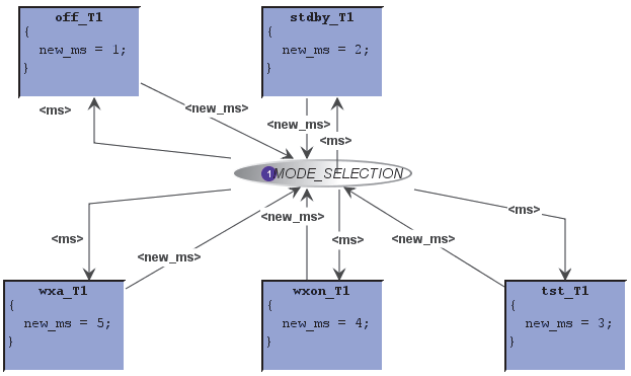


Figure 8. System model of the mode selection part of the WXR application

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.

Figure 6 presents a refinement of the design process presented in Figure 1 dedicated to automation design issues. As introduced before the design driver of the iterative process is the issue of performance of the couple tasks-system. Increasing automation, for instance by migrating functions from the tasks to the systems, is, at first glance, a very good candidate improving the performance. However, it is not an easy task to make explicit which tasks have to be migrated and how the system has to be modified in order to be able to perform the tasks previously performed by the users and represented in the task model.

The point of the paper is not to provide design guidelines for the design of automation i.e. which functions have to be migrated and how, but on the other side to demonstrate that the notations and their supporting tools are able to make explicit such evolutions.

Next sections present how some repetitive tasks (the testing of the weather radar) can be migrated to the system side and how such migration has a significant impact on the complexity of user's activities.

3.3 First analysis for automation design: task migration

When analyzing the task model in Figure 5, 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 10 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).

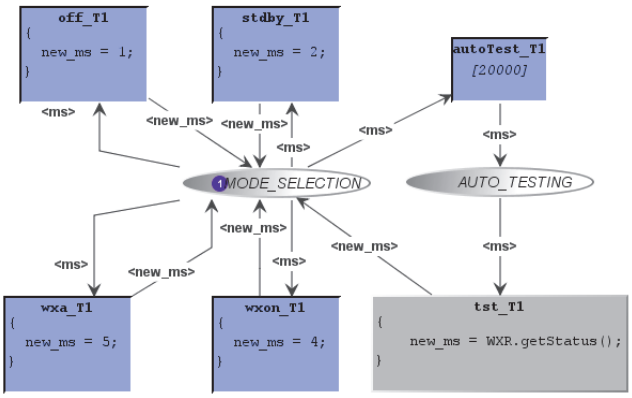


Figure 9. System model of the mode selection part of the WXR application with the automated testing

Figure 9 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 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 10.

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).

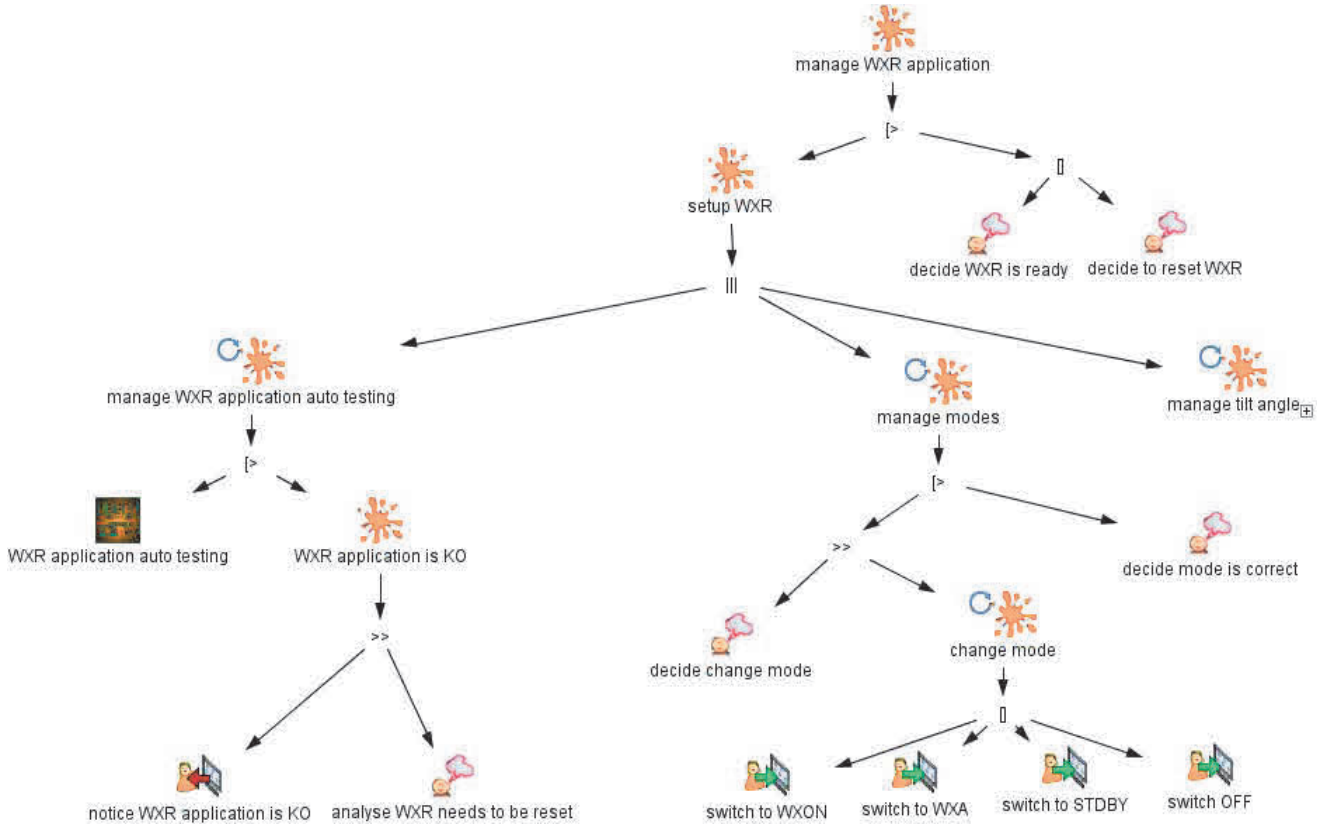


Figure 10. Task model of the manage WXR application activity when application testing has been automated

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 point was not to present here how to design more usable, reliable and safe interactive systems but to demonstrate that notations supporting a clear dichotomy between user's tasks and system functions make it possible to represent in a complete and unambiguous way allocation of function [2] and tasks migrations.

This work is the first step towards the definition of processes, notation and tools for assessing the performance of socio-technical systems featuring (partly-) autonomous behaviors. That work will take into account standard and erroneous behaviors both on the system side (usually called failures [27]) and on the user's side (usually called slips and mistakes [26]).

ACKNOWLEDGEMENTS

This work has been partly funded by R&T CNES Tortuga R-S08/BS-0003-029 and by Eurocontrol research network HALA! on Higher Automation Levels in Aviation.

REFERENCES

1. Barboni E., Ladry J-F., Navarre D., Palanque P., Winckler M. Beyond Modelling: An Integrated Environment Supporting Co-Execution of Tasks and Systems Models. In Proc. of EICS '10. ACM, 143-152.
2. Boy G. Cognitive Function Analysis for Human-Centered Automation of Safety-Critical Systems. Proceedings of ACM CHI 1998: 265-272
3. Cummings M.L., Bruni S., Collaborative Human-Automation Decision Making, Springer Handbook of Automation, pp. 437-447, 2009.

4. Diaper, D., Stanton, N. A. (eds.) The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, 2004. 650 p.
5. Johansson B., Fasth A., Stahre J., Heilala J., Leong S., Tina Lee Y., Riddick F., Enabling Flexible Manufacturing Systems by using level of automation as design parameter, Proc. of the 2009 Winter Simulation Conference, 13-16 dec. 2009.
6. Navarre, D., Palanque, P., Ladry, J., and Barboni, E. 2009. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Computer.-Hum. Interact. 16, 4 (Nov. 2009), pp. 1-56.
7. Navarre, D., Palanque, P., Winckler, M. Task Models and System Models as a Bridge between HCI and Software Engineering. "Human-Centered Software Engineering Models, Patterns and Architectures for HCI". Springer (HCI Series), 2009, pp. 357-385.
8. Palmer, E. "Oops, it didn't arm." - A Case Study of Two Automation Surprises . 8th International Symposium on Aviation Psychology, Ohio State University, 1995.
9. Paterno, F., Mancini, C. and Meniconi, S. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: Proc. of Interact'97. Chapman & Hall (1997), 362-369.
10. Parasuraman, R.; Sheridan, T.B.; Wickens, C.D. "A model for types and levels of human interaction with automation" Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Trans. on, vol.30, no.3, pp.286-297, May 2000.
11. Proud, R. W., Hart, J. J., & Mrozinski, R. B. (2003). "Methods for Determining the Level of Autonomy to Design into a Human Spaceflight Vehicle: A Function Specific Approach," Proc. Performance Metrics for Intelligent Systems (PerMIS '03), September 2003.
12. Sarter, N. D., Woods D. How in the World Did I Ever Get Into That Mode? Mode Error and Awareness in Supervisory Control, Human Factors, 37(1), (1995).
13. Bastide R., Palanque P. A Petri Net Based Environment for the Design of Event-Driven Interfaces. 16th International Conference on Application and theory of Petri Nets (ATPN'95), LNCS, Springer Verlag, Torino, Italy, 20-22 June 1995.
14. Ladry J-F., Navarre D., Palanque P. Formal Description Techniques to Support the Design, Construction and Evaluation of Fusion Engines for SURE (Safe Usable, Reliable and Evolvable) Multimodal Interfaces. In: ICMI-MLMI 2009, ACM, p. 135-142, 2009.
15. Bastide R., Palanque P. Modeling a Groupware Editing Tool with Cooperative Objects. Concurrent Object-Oriented Programming and Petri Nets. G. Agha, F. De Cindio (Eds.), Springer-Verlag, V. 2001, LNCS, 305-319.
16. Lakos, C, & Christensen, S. A General Systematic Approach to Arc Extensions for Coloured Petri Nets." 15th International Conference on Application and Theory of Petri Nets, ICATPN'94, Zaragoza. LNCS no. 815. Berlin, Springer (1994) 338-57.
17. Lakos, C. Language for Object-Oriented Petri Nets. #91-1. Department of Computer Science, University of Tasmania, 1991.
18. Ramamoorthy, C. V., and Ho, G. S. "Performance Evaluation of Asynchronous Concurrent Systems." IEEE Transactions of Software Engineering 6, no. 5 (1980) 440-449.
19. Palanque P., Ladry J-F, Navarre D., Barboni E. High-Fidelity Prototyping of Interactive Systems can be Formal too 13th International Conference on Human-Computer Interaction (HCI International 2009) San Diego, CA, USA.
20. Bastide R., Navarre D. & Palanque P. & (2003) A Tool-Supported Design Framework for Safety Critical Interactive Systems. Interacting with computers, Elsevier, Vol. 15(3), 309-328.
21. Suchman, L. A., Plans and situated actions: the problem of human-machine communication. 1987. 0-521-33739-9.
22. Martinie, C., P. Palanque, et M. Winckler. Structuring and Composition Mechanism to Address Scalability Issues in Task Models. LNCS INTERACT. Lisbonne, Portugal: Springer, 2011.
23. Card S.K., Moran T.P. & Newell A. The psychology of Human-Computer Interaction. Lawrence Elbaum Associates, 1983.
24. Norman D. The design of everyday things. MIT press 1998.
25. Martinie C., Palanque P., Barboni E. & Ragosta M. Task-Model Based Assessment of Automation Levels: Application to Space Ground Segments. IEEE International Conference on Systems, Man and Cybernetics, Anchorage, IEEE Computer Society, 2011.
26. J. Reason: Human Error. 1990. Cambridge University Press.
27. Avizienis A., Laprie J-C., Randell B., Landwehr C. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Trans. Dependable Sec. Comput. 1(1): 11-33, 2004.