Multi-Models-Based Engineering of Collaborative Systems: Application to Collision Avoidance Operations for Spacecrafts (2014)

Célia Martinie, Eric Barboni, David Navarre, Philippe Palanque, Racim Fahssi, Erwann Poupart, Eliane Cubero-Castan

▶ **To cite this version:**

Célia Martinie, Eric Barboni, David Navarre, Philippe Palanque, Racim Fahssi, et al.. Multi-Models-Based Engineering of Collaborative Systems: Application to Collision Avoidance Operations for Spacecrafts (2014). ACM SIGCHI conference Engineering Interactive Computing Systems (EICS 2014), ACM Special Interest Group on Computer-Human Interaction, Jun 2014, Rome, Italy. pp.95-94, 10.1145/2607023.2607031 . hal-04080938

HAL Id: hal-04080938
https://hal.science/hal-04080938

Submitted on 25 Apr 2023

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.

Open Archive TOULOUSE Archive Ouverte (OATAO)
OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : http://oatao.univ-toulouse.fr/
Eprints ID : 13035

To link to this article : DOI :10.1145/2607023.2607031
URL : http://dx.doi.org/10.1145/2607023.2607031

To cite this version : Martinie, Celia and Barboni, Eric and Navarre, David and Palanque, Philippe and Fahssi, Racim and Poupart, Erwann and Cubero-Castan, Eliane *Multi-Models-Based Engineering of Collaborative Systems: Application to Collision Avoidance Operations for Spacecrafts*. (2014) In: ACM SIGCHI conference Engineering Interactive Computing Systems - EICS 2014, 17 June 2014 - 20 June 2014 (Roma, Italy).

Any correspondance concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr

Multi-Models-Based Engineering of Collaborative Systems: Application to Collision Avoidance Operations for Spacecraft

Célia Martinie, Eric Barboni, David Navarre, Philippe Palanque, Racim Fahssi
ICS-IRIT, University of Toulouse
118, route de Narbonne
F-31062, Toulouse, France
{lastname}@irit.fr

Erwann Poupart, Eliane Cubero-Castan
CNES
Centre Spatial de Toulouse
18, avenue Edouard Belin
F-31401 Toulouse Cedex 9, France
{Firstname.Lastname}@cnes.fr

ABSTRACT
The work presented in this paper is based on a synergistic approach [1] integrating models of operators' tasks (described using the HAMSTERS notation) with models of the interactive system (described using the ICO notation) they are using. This synergistic approach makes it possible to bring together two usually independent (but complementary) representations of the same world. Even though supported by modeling and simulation tools, previous work in this area was rather theoretic focusing on concepts and principles in order to articulate this synergistic use of the models. The current article extends this line of research to address groupware applications. These extensions are performed on HAMSTERS notation in order to describe activities involving multiple users dealing with information flow, knowledge they are required to master and communication protocol (synchronous or asynchronous). Other extensions are performed on PetShop tool (supporting the ICO notation) in order to model and execute local and distant groupware applications. These extensions have been brought together by a more complex synergistic module bringing the two views together. Lastly, these extensions have been used for the modelling, design, and construction of a groupware system dedicated to collision avoidance of spacecraft with space debris. This case study is used to assess the applicability of the contributions and to identify paths for future work.

Author Keywords
Task and interactive systems models, CSCW, groupware, space systems.

INTRODUCTION
The evolution of computer use from one computer for several persons to many computers for one person could have been the end of multi-user computing. However, the widespread of internet and the rise of social computing has demonstrated that dealing with single user applications is nowadays part of history. Designing interactive systems thus requires most of the time to need to address the needs of group of users involved in common tasks for which the communication, cooperation and production is mediated by computers. Despite this undeniable situation, most of the research contributions in the area of interactive systems engineering still focus on single user applications. This is easily understandable as multi-users application are far more difficult to build than single user ones. This difficulty comes from different sources:

- The difficulty to gather and understand the requirements as well as the need of the users;
- The difficulties to address the required communication infrastructures in order to allow both synchronous and asynchronous communication between the users;
- The difficulty to ensure usability of these applications that are used jointly by different users (with different characteristics and needs) and under different environmental conditions (time zones, seasons, light, sound, …);
- The difficulty to ensure the reliability of these computing systems involving underlying communication mechanisms, networks…

This paper aims at proposing a model-based approach for the design of usable and reliable collaborative applications.

To address the usability issue we propose a notation for describing collaborative task i.e. tasks having group of users trying to achieve common goals. This notation extends current models such as GTA [25] or CTT [18]. As for CTT, which the most mature notation in that domain, extensions refine further the task types (see section 3), adds explicit representation of information and knowledge

required for performing the tasks and does not require the construction of an "artificial" task model describing the collaboration.

To address reliability, we propose the use of the ICO formalism and its related tool PetShop extended in order to edit and execute models of interactive distributed applications. This work takes advantage of previous work done with ICO notation to formally specify distributed applications over Corba middleware [2]. Following the philosophy presented in [1] we propose also a synergistic approach integrating models of operators' tasks (described using the extended HAMSTERS notation) with models of the interactive system (described using the ICO notation).

These various elements are successively presented in the paper. This presentation is followed by the description of the application of the approach on a real life case study from the space domain. This case study consisted in designing and modeling a collaborative collision avoidance management application for the CNES (French Space Government Agency) Orbit Computation Center. Current existing and in use applications are not supported by dedicated tools for collaboration. They are distributed over many time zones, involve multi-national teams and aim at forecasting and avoiding collisions between spacecraft and space debris.

RELATED WORK

In the field of Computer Supported Cooperative Work (CSCW) and groupware, many contributions deal with classification and properties of groupware applications [20], but also with guidelines about how to design and evaluate the usability of such applications [6]. Another thread of work addresses user interfaces, interaction techniques and underlying computer-based tools for supporting collaborative activities [23].

The engineering of groupware applications is also represented amongst the scientifics contributions. For example, Jakobsen et al. propose a collaborative development environment for software development [11] while Bates et al. propose an applicative framework for integrating collaborative functionalities [3]. Wu and Graham [26] propose an architectural design framework for software architectures of collaborative applications based on the concept of workspace while Greenberg et al. [22] propose a toolkit for the construction of real-time groupware.

Model-based approaches for engineering groupware applications can be of several types. Workflow approaches such as YAWL [24] target at describing and simulating workflow activities systems and subsystems while some recent approaches such as BPEL4People try to take into account user's activities [10]. Task-model based approaches target at describing collaborative activities involving group of users collaborating to achieve a common goal [25] [18] [12]. UML-based and tool supported approaches have also been proposed [8] to support development of collaborative applications. However, while the firstly mentioned contributions focus on user interface and interaction techniques and the later ones on workflows and activities of group of users, none of them propose a solution to bring those two worlds together and address in one single framework these two threads of work that are required to develop usable groupware. Indeed, while efficiency of interaction can be addressed through user interface design, the effectiveness of communication tools and interaction techniques can only be addressed by exhaustive representation of tasks to be performed by collaborating users.

AN INTEGRATED ENVIRONMENT SUPPORTING THE CO-EXECUTION OF TASKS AND GROUPWARE

We propose a synergistic use of task and system models for ensuring consistency, coherence and conformance between collaborative activities and their associated distributed user interfaces and applications. In order to reach this research goal, several extensions that have been made to an existing framework previously introduced in [1]. The presented framework and tool suite is composed of:

- A modeling and software development environment for interactive systems (Petshop).
- A modeling and simulation CASE tool for engineering user tasks (HAMSTERS).
- A synergistic module for linking system behavioral models and user task model, which then enables their co-execution (Synergistic module).

Petshop extensions

The very early versions of Petshop tool followed a Corba approach to handle communication between models [2]. Naming Service and Interface Repository were mandatory parts of Corba to make communication possible. But as Corba implies a time consuming workflow (IDL compilation, IDL registration), the Naming Service and Interface Repository were centralized and then the prototyping activities were complex.

As Petshop is Java based, we naturally studied Java RMI in order to reduce the compilation phase (as it was embedded in Java). However, the centralized registration issues remained. In order to ease deployment and more flexibility, the communication layer was updated to follow a peer to peer approach, where each Petshop instance is serving models they run. The proposed implementation uses the Ivy bus. The Ivy protocol[1] was initially designed for broadcasting text messages using regular expressions (regex) for prototyping Air Traffic Control applications. As the approach is using regex, it makes possible to deploy lightweight clients on separate computers. Those separate

[1] http://www.eei.cena.fr/products/ivy/documentation/

computers are not able to detect some specifics messages on the bus, which makes it possible to log workflows without polluting the client computers. One of the main advantages of the Ivy approach is that its adoption didn't require modifying the ICO notation which was originally designed for describing systems as a set of cooperating instances of classes (independent from their location).

HAMSTERS extensions

The HAMSTERS notation and CASE tool has been introduced in 2010 in order to provide support for task-system integration at the tool level [1]. Since then, this tool and notation has been refined several times in order to provide support for:

- Automation design. The notation has been extended to help with the analysis of function allocation between human and system thanks to the refinement of cognitive tasks into analysis and decision subtypes of cognitive tasks according to the Parasuraman model of human information processing [13].
- Structuring a large number and complex set of tasks introducing the mechanism of subroutines [16].
- Precise description of knowledge, information and objects required and manipulated [15] in order to accomplish tasks.

These elements are necessary to describe collaborative activities but they are not sufficient. Hereafter are the extensions we propose in order to deal with groupware.

Adding notation elements to describe collaborative activities

Collaborative work is performed by several persons, each one having a role in the achievement of common goals. The concept of role we are using is the same as the one used in [18] and [25]. Figure 1 illustrates the structure of a HAMSTERS project with several roles, each of them having associated task models structured according to the mechanism of subroutines introduced in [16]. In the same way, we also integrate the concept of actor [25] in the HAMSTERS notation and tool.

Collaborative work can be described at different abstraction levels: at the group level and at the individual level. A group task is a set of task that a group has to carry out in order to achieve a common goal [17], whereas a cooperative task is an individual task performed by a person in order to contribute to the achievement of the common goal [21].

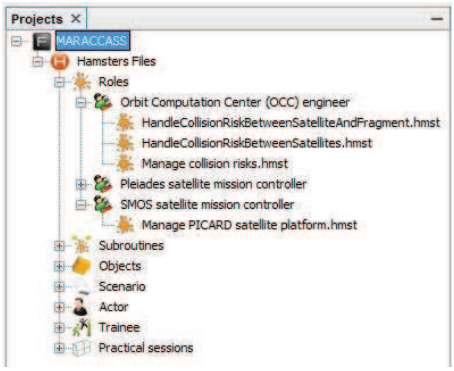

Figure 1. Structure of a HAMSTERS project

In order to be able to describe group tasks, we introduce several new task types illustrated in Figure 2 (in the last right column). These group tasks provide support for describing high level activities that a group of person have to accomplish:

- An abstract group task is a task that can be decomposed into user, system, interactive and collaborative tasks.
- A group (of users) task is task that can be decomposed in user and collaborative user tasks.
- An interactive group task can be decomposed in interactive and collaborative interactive tasks.

		Abstract	Input	Output	I/O	Processing	Group
Abstract			Not applicable	Not applicable	Not applicable	Not applicable	
User	Indiv.						
	Coop.						
Interactive	Indiv.					Not applicable	
	Coop.					Not applicable	
System							

Figure 2. Task types in HAMSTERS

- A system group task can be decomposed in system tasks.

The refinement of group tasks into low-level activities needs fine-grain task types to describe individual and cooperative tasks that have to be performed in order to contribute to the group activities. As individual task types were already available within HAMSTERS, we then introduce cooperative tasks, illustrated in Figure 2. A cooperative task is a task related to a role and accomplished in correlation with another cooperative task that relates to a different role. A cooperative task may be of various types within the user and interactive main family types.

Cooperative tasks may be performed within various space-time constraints (local/distant, synchronous/asynchronous) [7]. These constraints can be described with notation elements illustrated in Figure 3.

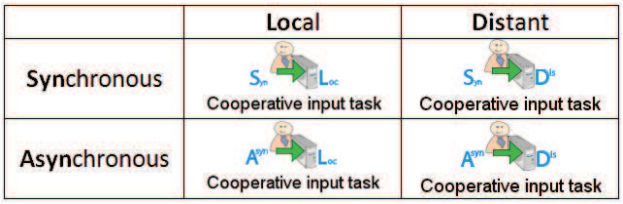

Figure 3. Elements of notation related to space-time constraints

Cooperative task may be dedicated to one or more of the following type of collaborative activities: production, coordination, communication. It is then possible to associate one or more properties amongst this set. For example, Figure 4 a) shows that one task is dedicated to coordination whereas Figure 4 b) shows that the task is dedicated to both coordination and communication.

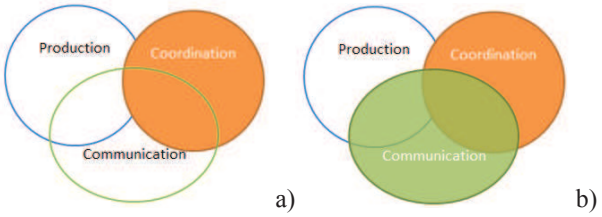

Figure 4. Example of cooperative task properties from a "functional clover" [9]

Adding edition and simulation capabilities

Several functionalities have been added to HAMSTERS CASE tool[2] in order to support the edition and simulation of tasks models describing collaborative activities:

- At the edition level, a mechanism to link cooperative tasks across task models belonging to different roles (by mean of contextual menu appearing after a right click on a cooperative task and proposing the list of available cooperative tasks in the other task models belonging to the other roles, as illustrated in Figure 5).

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

- At the simulation level, a mechanism to enable the execution of several models belonging to different roles.

HAMSTERS make it possible to structure users' activities belonging to one role in several tasks models edit several task models. The association between cooperative tasks of different roles is not made across an additional task model (as for example the cooperative task model in CTTe [18]) but it is achieved via an internal correspondence table in the tool.

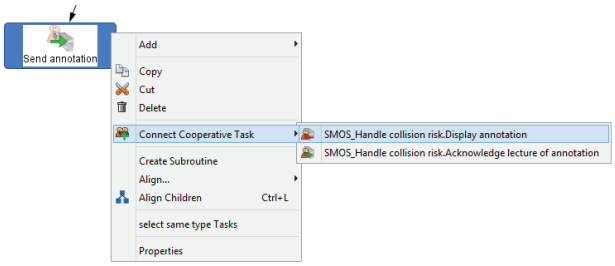

Figure 5. Defining correspondence of cooperative tasks of 2 task models representing different roles

Synergistic module extensions

The synergistic module has been improved for both design time editing of correspondences and runtime simulation to support collaborative activity modelling and simulation.

Design time

To support collaborative simulation, the editing of correspondences requires three main improvements w.r.t what have been previously done [1]:

- Allowing the use of several task models instead of a unique one.
- Allowing the use of several collaborative ICO models instead of models concerning a local instance of the application.
- Taking into account cooperative input and output tasks, additionally to input and output tasks previously used.

Runtime

The two main improvements for the runtime use of models concern the task driven simulation as it mainly requires the use of HAMSTERS:

- We added means to select the HAMSTERS role involved in the simulation.
- An important improvement is about the value selection for task objects: executing an input task may requires to select a particular set of values that may be found within the PetShop data set or that may be provided by the user (for instance feeding in a text box). These values are thus used to control the PetShop execution of a transition.

Architecture of the extended environment

As stated in [1], the integration of the tools relies on specific API provided by both HAMSTERS and PetShop. In this

section we present how the whole tool suite has been improved to handle synergistic execution.

As described in the previous sections, the two main specificities introduced by this new approach are:

- HAMSTERS allows the editing and simulation of several roles described using several task models.
- PetShop allows the editing and execution of several distributed ICO models.

The goal of the tool suite is thus improved in order to allow the specification of a distributed user application and the related user activities. The whole interactive system may be considered as a set of pairs of single users and single user application with communications means supported by both the applications and the users. Making these two parts of a pair correspond requires identifying points of connection for both design time and runtime:

- From each tasks specification we extract the set of interactive tasks and cooperative interactive tasks (for both input and output) representing a set of manipulations that can be performed by the users on the system and outputs from the system to the users.
- From each ICO specification we extract the activation and rendering function that may be seen as the set of inputs and outputs of the system model.

Design time architecture

The principle of editing the correspondences between the models of one pair {*role, system*} is to put together interactive and cooperative interactive input tasks (from the task models) with system inputs (from the system models) and system outputs (from the system models) with interactive and cooperative interactive output tasks (from the task models). Setting up this correspondence may show inconsistencies between the task and system models such as interactive tasks not supported by the system or rendering information not useful for the task performance. The correspondence editing process is presented on Figure 6 where each tool feeds the correspondence editor with information from the API in order to notify it with modifications are done both in the task model and in the system model.

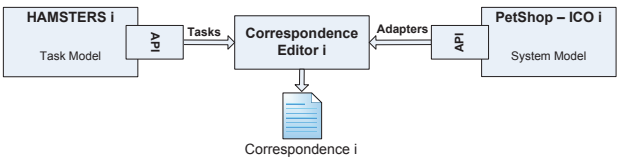

Figure 6. Design time architecture of the tool suite.

This process that produces a correspondence is repeated for each pair on the entire system. Each pair is being put into correspondence separately producing a set of independent correspondences. These dependencies totally rely on the connection of task tree leaves (both input and output cooperative tasks). These elementary tasks are performed on the local system (represented by a set of ICO models). The separation of the correspondences relies on the fact that each part of the distributed system is modelled by a set of independent ICOs, connected by communication means (method calls) supported at runtime by the Ivy bus.

As the correspondence editing goal is to put together user activities and interactive features, this correspondence is only possible with the finest grain item of both descriptions.

Runtime architecture

Our framework allows the co-execution of task and system models controlled by both the execution of the system model and the execution of the task model as shown in Figure 7 (where the correspondences are those produced by the process described by Figure 6).

Figure 7 highlights the two ways communication within each pair *{task, system}* allowed by the services embedded within the two following APIs:

- Between HAMSTERS and the Simulation Controller: On one side HAMSTERS notifies changes in the current scenario to the Simulation. On the other side the Simulation Controller is able to ask to perform the corresponding task (according to the correspondence provided by the Correspondence editor), simulating the user action.
- Between PetShop and the Simulation Controller: on one side the PetShop interpreter notifies the Simulation Controller the evolution of the current execution of the system model (notifications come from both rendering and activation functions). On the other side, the Simulation Controller fires the corresponding activation adapter (according to the correspondence provided by the Correspondence editor) simulating the user action.

Such as in [1] such architecture allows a two ways simulation controlled by the system execution or controlled by the task simulation. The principle of the simulation is the same as in our previous work:

- When driven by the tasks, building a scenario using HAMSTERS is translated by the Simulation Controller into user actions within PetShop (a sequence of transition firing).
- When driven by the system execution, user actions are directly linked to the corresponding tasks from the task model and the user's action on the user interface of the application change the current state of the task model simulation.

The task driven simulation introduces the data correspondence. This correspondence is built on runtime capabilities of the two tools HAMSTERS and PetShop:

- HAMSTERS provides the description of objects, information and knowledge required and manipulated in order to accomplish tasks.

- PetShop, by describing data structure and data flow of the system, provides the set of data manipulated at any moment by any actions (firing of transitions).

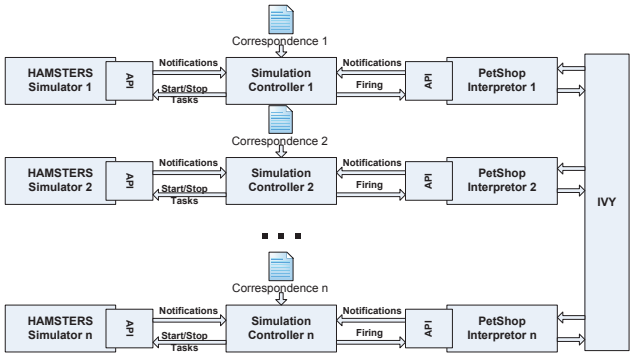

Figure 7. Runtime architecture of the tool suite

It is thus possible for any couple (*task, event handler*) within the correspondence description to match system data and the required objects for the task performance. An example of such data matching is provided by the case study (see next section). The originality of this work relies on the distribution of the system. The choice made to handle this distribution (as illustrated by Figure 7) is to use the communication means offered by PetShop (thanks to the Ivy software bus). The distribution is thus fully supported by the system side of the specification as it requires concrete data communication that should be out of the scope of task modelling.

EXAMPLE FROM A LARGE CASE STUDY

The proposed tool suite has been used to design and develop a prototype of groupware application belonging to the space ground segment category of applications. This study has been led in the context of a Research and Technology project funded by the French Space Government Agency (CNES). MARACCASS stands for Models and Architectures for the Resilience and Adaptability of Collaborative Collision Avoidance System for Spacecraft and aims at studying methods, techniques and tools to design and develop collaborative applications. This project is particularly targeting groupware for the management of collision avoidance between satellites and space objects. In this section, we present illustrative extracts from the case study which are relevant to highlight the key points of the contribution.

Management of collision risks between space objects and satellites

CNES and various other international agencies have to cope with the increasing number of space fragments, which are a threat to on-going satellite missions. Collision avoidance management is a collaborative, cross-team, and international activity. Amongst the national and international organizations, two main types of teams can be distinguished: the space observation teams and the satellite mission teams. The observation teams, thanks to various equipment's and tools are gathering information about space objects and their trajectories (past, present, future). The mission teams focus on one particular space object (usually a satellite) and are monitoring and controlling the space object they are in charge of and its operations. If the observation team detects a collision risk between a satellite and a space object, it contacts and alerts the mission team in charge of the satellite.

Roles and main goals to manage collision risks

In this case study, we take the example of the collaboration between the CNES team in charge of monitoring space objects (called the Orbit Computation Center or OCC) and the SMOS[3] satellite mission team. In order to collaboratively manage a collision risk, the teams are assisted with several non-integrated software tools: individual software tools to analyze probability of collision and traditional communication tools (email and telephone) to coordinate and communicate about the risk.

Preliminary work before high-fidelity prototyping phase

The first phase of the project has consisted in analyzing current activity with the production of corresponding task models. Then, we proposed several low-fidelity prototypes for a new groupware application to support collaborative activities of collision risk management. These low-fidelity prototypes take into consideration groupware principles [7] but also contributions about design considerations for collaborative visual analytics [9]. We then produced task and system models from low-fidelity prototypes that had been validated with operational teams.

In the next paragraphs we present extracts from models and from the high-fidelity groupware prototypes that highlight how the proposed framework has been applied to develop a high-fidelity prototype of the collaborative application for collision risk management. In these extracts, we will focus on the collaborative asynchronous activities related to posting annotations (OCC engineer role) and consulting these annotations (SMOS controller role) in the corresponding remote applications. Figure 8 and Figure 9 presents screenshots of the two remote applications dedicated to collaborative management of collision risks. Figure 8 presents the application dedicated to OCC engineers (with a larger set of functionalities such as deep probabilistic calculus and Conjunction Summary Messages creation and edition). In the presented screenshot, a popup window is opened in order to let the OCC engineer edit an annotation. Figure 9 presents the application dedicated to the mission controllers with a reduced set of functionalities. Its main purpose is to provide situation awareness about the collision risks related to the mission and communication and coordination support. In the presented screenshot, an

[3] http://smsc.cnes.fr/SMOS/index.htm

annotation is displayed (pined to the table) to the attention of the SMOS mission controller.

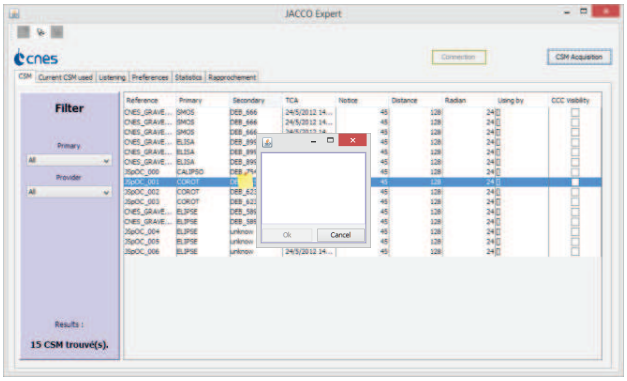

Figure 8. Screenshot of the Hi-Fi prototype for collision risks management dedicated to OCC engineers

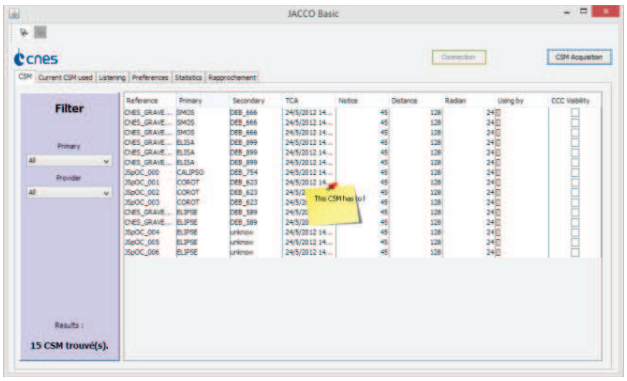

Figure 9. Screenshot of the Hi-Fi prototype for collision risks awareness dedicated to SMOS mission engineers

Task models

In this section, extracts of the task models illustrate the HAMSTERS extensions and especially how the new cooperative task types have been applied to support the development of the groupware Hi-Fi prototype.

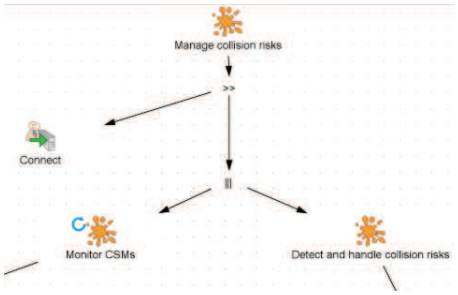

Figure 10. Extract of the highest level task model for the OCC activities (OCC engineer role)

Orbit Computation Center (OCC) engineers

Figure 10 presents an extract from the set of activities that have to be performed by the expert engineer on duty from the Orbit Computation Center to monitor and manage collision risks.

Figure 11 presents an extract from the set of activities performed once a collision risk has been detected for the SMOS satellite mission. In particular, it shows the sequence of activities led when the SMOS mission controller was not available for a live communication. The OCC engineer first creates an annotation ("Create annotation" input tasks), then positions the annotation (iterative task "Move annotation") until the position is adequate ("Fix annotation position" input task). The OCC engineer then edits the annotation (input task "Edit annotation"), decides to send the annotation (cognitive decision task "Decide to send annotation") and then send the annotation (cooperative asynchronous task "Send annotation").

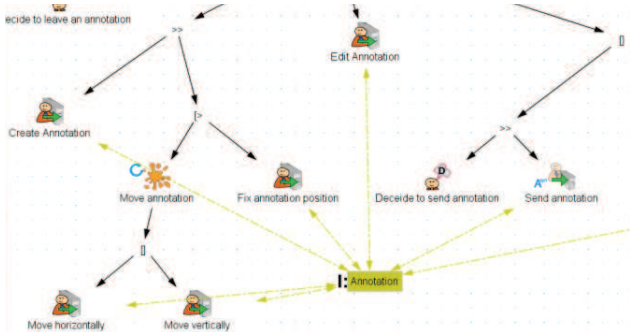

Figure 11. Extract of the task model "Handle collision risk between satellite and fragment" for the OCC engineer role

SMOS command and control room controllers and engineers

Figure 12 presents an extract from the set of activities that have to be performed by the SMOS controller when warned by the OCC engineer.

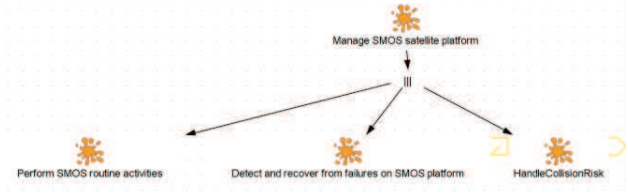

Figure 12. Extract of the highest level task model for the SMOS mission activities

Figure 14 presents an extract from the set of activities performed once a collision risk has been detected. In particular, this set of activities is cooperative and bound to the above presented set of activities for the OCC engineer role. Once the OCC engineer has sent an annotation, it is displayed in the SMOS remote application (cooperative output task "Display new annotation"). When the SMOS mission controller will be available for consulting the application, s/he detects and acknowledges reception of the annotation (cooperative input asynchronous task "Acknowledge lecture of annotation"). S/he then analyzes the reported risk and may delete the annotation (cooperative input task "Delete annotation").

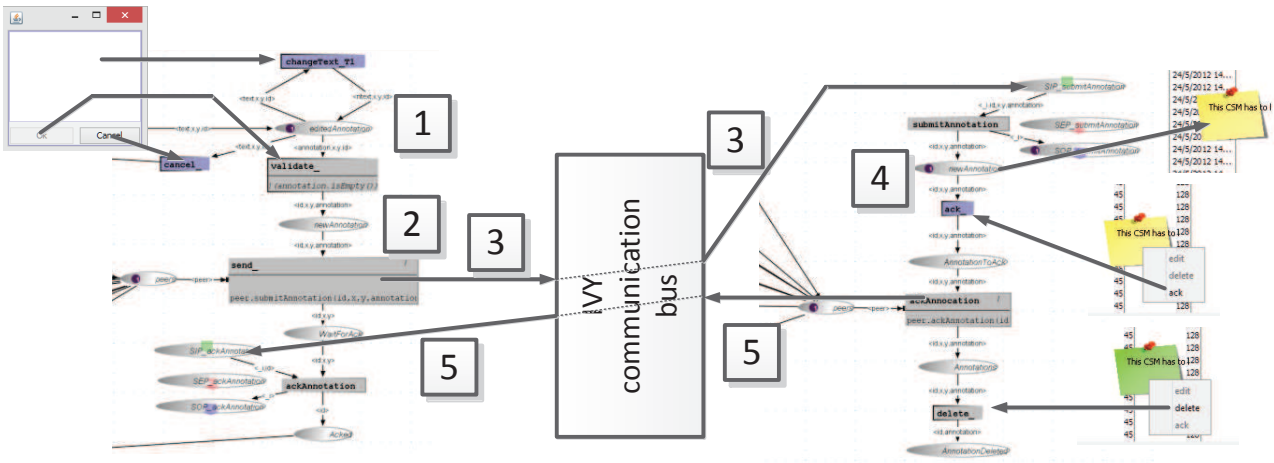

Figure 13. ICO modelling of the two remote applications COO (left part) and SMOS (right part)

ICO modelling of the collaborative application to manage collision risks

Figure 13 illustrates an excerpt of the connected models of the two remote applications OCC (left part of the figure) and SMOS (right part of the figure). The two excerpt of the models presented here describe how the editing of an annotation is performed on the OCC side and how it is sent and acknowledged back on the SMOS side. In both case, a snapshot of the dedicated graphical part is provided. These models behave as follows:

1. When in editing mode (a token is put in place EditedAnnotation at the top of the left part of Figure 13), it is possible to edit the text of the annotation using the edition window (see left part of Figure 13). It results in a state change (by the firing of transition changeText_, changing the marking of place EditedAnnotation with the new text) or it is possible to validate or cancel the editing (where button OK fires the transition validate_ and button CANCEL the transition cancel_).
2. When validated (cancellation producing a return to the initial state) a token is put in place newAnnotation, the created annotation is sent to the peer application (SMOS) by the remote invocation peer.submitAnnotation(…).
3. The communication bus Ivy then sends this method call to the ICO SMOS model (a token is put in the service input port of the corresponding method, the place SIP_SubmitAnnotation on the top of the right part of Figure 13).
4. The firing of the following transition submitAnnotation puts a token in place newAnnotation making the transition ack_ available (which is translated into activating the corresponding menu item within the popup menu show on the right part of Figure 13).
5. Using the menu item ack results in setting a token in place AnnotationToAck which thus allows the firing of transition ackAnnotation. The firing of this transition results in the remote method peer.ackAnnotation(…) invocation (on the bottom of the left part of the figure) and finally makes the transition delete_ fireable (the corresponding menu item becoming enabled).

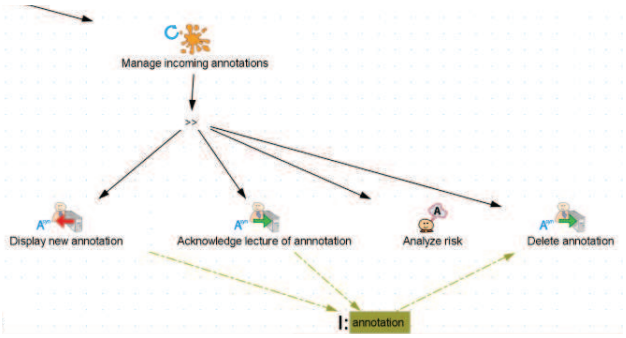

Figure 14. Extract of the task model "Handle collision risk" for the SMOS mission controller role

Distributed co-execution between tasks and applications

Figure 15 illustrates two remotes instances of the tool suite presented in this paper, connected using the Ivy communication bus. The left part (resp. the right part) corresponds to the editing of the OCC (resp. SMOS) models. The left-hand part of the figure shows the project structure of the tool suite (i.e. a set of modules of the NetBeans IDE[4]). This tool may be divided into four parts:

- The top part is a set of classical IDE menu bars and tool bars buttons.
- The left part provides means to navigate amongst the project files (java sources, ICO, HAMSTERS and correspondence models).
- The right part shows properties of the sources or models (bottom part) and tools to modify the currently selected model (on the top part a specific toolbox appears

[4] https://netbeans.org/features/index.html

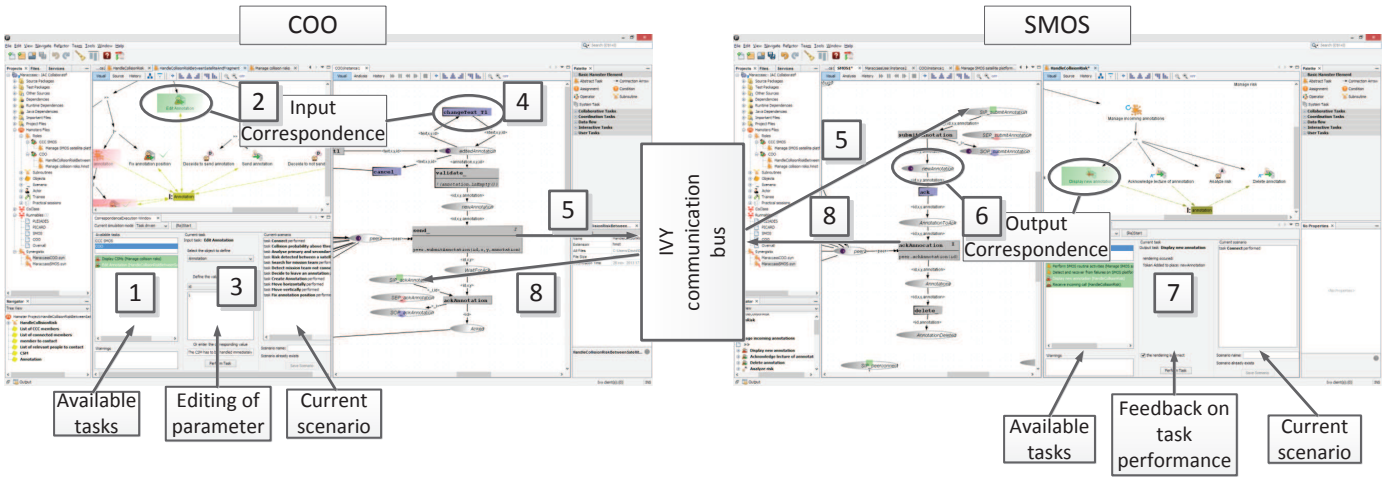

Figure 15. Illustration of the collaborative co-execution using two instances of the tool suite

depending on the king of the selected model (HAMSTERS or PetShop).

- The center part allows the editing and execution control of both the sources and models. The layout of this part is fully reconfigurable as illustrated by the layout difference of the left and right part of the figure.

To illustrate the synergistic exploitation of both the system and task models, we use the models presented in the previous sections, following the same scenario as used for the ICO models presentation (editing of an annotation). As the correspondence editing between system and task models are a simple table editing and was already presented in [1], we only focus here on the task driven simulation with a special focus on the cooperative activities.

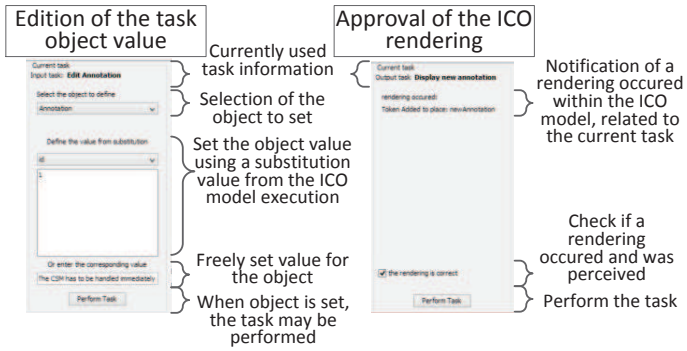

Figure 16. Detailed view on task-system related data.

Following the eight numbered steps presented on the figure, the behavior of the task driven simulation is the following one:

1. A set of available tasks is provided by the HAMSTERS environment that is selectable within the associated list box.
2. The selected task is connected to a transition by the correspondence editing.
3. As the tasks requires an object (see task Edit Annotation in Figure 11 in Task models section of the case study), a dedicated panel is available (see the left part of Figure 16 for a zoomed snapshot of this part of the tool), providing means to select values built from the data (system side) of the running ICO model, related to the transition connected to the task. It is also possible to manually provide a value using the bottom text field of this panel (see bottom right part of Figure 16).
4. When set, the selected value is sent to the ICO model by sending an event (such as graphical part would have done it).
5. Such as in the step 3 of Figure 13 (the scenario described in the ICO models section), an invocation is performed on the remote application.
6. When a token enters a place within the ICO model, a rendering may occur and this rendering may be related to an output HAMSTERS task, using the output correspondence edition.
7. When an output task is selected, a dedicated panel appears at the bottom of the tool, showing whether a rendering occurs and if it corresponds to an output correspondence (Figure 16 shows a detailed view of this panel). In this panel it is possible to indicate if the rendering was effectively correct and perceive (for log purpose).
8. Finally, such as in step 5 of Figure 13 (the scenario described in the ICO models section), an invocation is performed back on the remote application.

CONCLUSIONS AND FUTURE WORK

This paper has proposed a tool supported approach for bridging the gap between tasks and system views in the design of multi-user interactive systems. To this end we have briefly introduced extensions to the notation called HAMSTERS for the description of multi-users tasks

models. These extensions allow the production of very detailed description of collaborative tasks beyond the expressive power of other task notations. This expressive power has allowed us to embed this notation within a synergistic framework where collaborative task models are connected to the interactive parts of a distributed system.

Through the application of the approach on a real life case study (that was carried out over the 3 years lifespan of the project) we have demonstrated the validity of the approach in the context of space critical systems. However, this validation has only demonstrated that the notations are able to describe multi-user activities and interactive systems and that they were able to scale enough to describe a real life application. We have not presented in this paper all the work that has been carried out around the definition of the user interfaces, the interaction techniques and the communication and collaboration functionalities. This work has been performed using a User Centered Design approach based on low-fidelity and high-fidelity prototypes.

The work presented here belongs to a longer term research program targeting at the design of resilient interactive systems using model-based approaches. Future work targets at exploiting this approach, to propose model-based usability evaluation of multi-user interactive systems extending the approach proposed for mono-user ones in [4]. We will also build on this model-based work to identify function allocations between operators and interactive systems to design more usable and reliable automation for critical systems.

ACKNOWLEDGMENTS

This work was partly sponsored by CNES R&T MARACCASS. We would like to thank the OCC team and in particular Jean-Claude Agnèse and François Laporte.

REFERENCES

1. Barboni E., Ladry J-F., Navarre D., Palanque P. and Winckler M. Beyond modeling: an integrated environment supporting co-execution of tasks and systems models. EICS'10, 165-174.
2. Bastide, R., Palanque, P., Sy, O., Navarre, D. Formal specification of CORBA services: experience and lessons learned. In Proc. of OOPSLA 2000, 105-117.
3. Bates, J.; Spiteri, M.D.; Halls, D.; Bacon, J. Integrating real-world and computer-supported collaboration in the presence of mobility. In Proc. 7th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1998, 256-261.
4. Bernhaupt R., Navarre D., Palanque P., Winckler M. Model-Based Evaluation: A New Way to Support Usability Evaluation of Multimodal Interactive Applications. Maturing Usability: Quality in Software, Interaction and Quality. Springer Verlag, April 2007.
5. Calvary, G., Coutaz, J., Nigay, L. From single-user architectural design to PAC*: a generic software architecture model for CSCW. In Proc. of CHI '97. ACM, NY, USA, 242-249.
6. Cockburn, A., Jones, S. Four principles for groupware design. Interacting with Computers. IJHCI, 7(2), 1995, 195-210.
7. Ellis C. A., Gibbs S. J., Rein G., Groupware: some issues and experiences, Comm. of the ACM, v.34 n.1, p.39-58, Jan. 1991.
8. Giraldo, W.J., Molina, A.I., Collazos, C.A., Ortega, M., Redondo, M. A Model Based Approach for GUI Development in Groupware Systems. In Groupware: Design, Implementation, and Use, LNCS, Vol. 5411. Springer-Verlag, Berlin, Heidelberg, 324-339.
9. Heer; J., Agrawala, M. 2007. Design Considerations for Collaborative Visual Analytics. Proc. of IEEE Symp. on Visual Analytics Science and Technology (VAST '07). IEEE Computer Society, 171-178.
10. Holanda, J. Merseguer, G. Cordeiro, and A. Serra. Performance evaluation of web services orchestrated with ws-bpel4people. Int. Journal of Computer Networks Communications, 2:18, 11/2010
11. Jakobsen M.R., Fernandez R., Czerwinski M., Inkpen K., Kulyk O.A., Robertson G.G. WIPDash: Work Item and People Dashboard for Software Development Teams. INTERACT 2009, pp.791-804.
12. Jourde F., Laurillau Y. & Nigay L. 2010. COMM notation for specifying collaborative and multimodal interactive systems. In Proc. of EICS '10. ACM, New York, NY, USA, 125-134.
13. Martinie C., Palanque P., Barboni E., Ragosta M. Task-Model Based Assessment of Automation Levels: Application to Space Ground Segments. Proc. of the IEEE SMC, Anchorage, 2011.
14. Martinie C., Palanque P., Navarre D., Winckler M. and Poupart E. Model-Based Training: An Approach Supporting Operability of Critical Interactive Systems: Application to Satellite Ground Segments, Proc. of EICS 2011, pp. 141-151, ACM DL.
15. Martinie, C., Palanque, P., Ragosta, M and Fahssi, R. Extending Procedural Task Models by Explicit and Systematic Integration of Objects, Knowledge and Information, ECCE 2013, 23-33.
16. Martinie, C.; Palanque, P. A. and Winckler, M. (2011): Structuring and Composition Mechanisms to Address Scalability Issues in Task Models. Proc. INTERACT (3) p. 589-609.
17. McGrath J. E. Groups: Interaction and Performance. Prentice Hall, Inc., Englewood Cliffs, 1984.
18. Mori, G., Paternò, F., Santoro C. 2002. CTTE: support for developing and analyzing task models for interactive system design. IEEE Trans. Softw. Eng. 28, 8 (August 2002), 797-813.
19. Paternò F., Ballardin G.: RemUSINE: a bridge between empirical and model-based evaluation when evaluators and users are distant. Interacting with Computers 13(2): 229-251 (2000)
20. Rama, J., Bishop, J. A survey and comparison of CSCW groupware applications. In Proc. of SAICSIT '06, 2006, Republic of South Africa, 198-205.
21. Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In C. E. O'Malley (Ed.), Computer-supported collaborative learning (pp. 69-197).
22. Roseman M, Greenberg S: Building Real-Time Groupware with GroupKit, a Groupware Toolkit. ACM Trans. Comput.-Hum. Interact. 3(1): 66-106 (1996).
23. Sun, A., Sun, C. Xpointer: an x-ray telepointer for relaxed-space-time wysiwis and unconstrained collaborative 3d design systems. In Proc. of CSCW '13. ACM, NY, USA, 729-740.
24. Van der Aalst, W.M.P, ter Hofstede, A.H.M. YAWL: Yet Another Workflow Language. Information Systems, 30(4):245-275, 2005.
25. Van der Veer, G. C., Lenting, V. F., Bergevoet, B. A. GTA: Groupware Task Analysis - modeling complexity. Acta Psychologica, 91, (1996), 297-322.
26. Wu, J., Graham, N. T. C: Toward Quality-Centered Design of Groupware Architectures**.** EHCI/DS-VIS 2007, 339-355.