Model-based dynamic distribution of user interfaces of critical interactive systems

David Navarre, Célia Martinie, Philippe Palanque, Alberto Pasquini, Martina Ragosta

Model-Based Dynamic Distribution of User Interfaces of Critical Interactive Systems

David Navarre, Célia Martinie, Philippe Palanque
Institute of Research in Informatics of Toulouse
University of Toulouse
Interactive Critical Systems (ICS) team
118, route de Narbonne
31042 Toulouse Cedex 9, France
{martinie, navarre, palanque}@irit.fr

Alberto Pasquini, Martina Ragosta
Deep Blue
Piazza Buenos Aires 20
00198 Roma, ITALY {alberto.pasquini,
martina.ragosta}@dblue.it

ABSTRACT

Evolution in the context of use requires evolutions in the user interfaces even when they are currently used by operators. This paper proposes a model-based approach to support proactive management of context of use evolutions. By proactive management we mean mechanisms in place to plan and implement evolutions and adaptations of the entire user interface (including behaviour) in a generic way. This generic model-based approach is exemplified on a safety critical system from the space domain. It presents how the new user interfaces can be generated at runtime to provide a new user interface gathering in a single place all the information required to perform the task. These user interfaces have to be generated at runtime as new rocedures (i.e. sequences of operations to be executed in a semi-autonomous way) can be defined by operators at any time in order to react to adverse events and to keep the space system in operation. Such contextual, activity-related user interfaces complement the original user interfaces designed for operating the command and control system. The resulting user interface thus corresponds to a distribution of user interfaces in a focus + context way improving usability increasing efficiency and effectiveness.

Author Keywords

Model-Based approaches, formal description techniques, interactive software engineering, automation, distributed user interfaces, dynamic reconfiguration of user interfaces.

ACM Classification Keywords

D.2.7 Distribution, Maintenance, and Enhancement, D.2.11 [Software] Software Architectures - Languages (e.g., description, interconnection, definition), H.5 [Information Systems] Information Interfaces and Presentation

General Terms

Design, Automation, Reliability, Human Factors.

INTRODUCTION

In the early days, the basic design rationale for User Interfaces for control rooms was to assign one display to each component to be monitored and one physical input to each command to be sent to one component of the controlled system. This resulted in very large command and control rooms being rather easy to design and build but rather cumbersome to operate. Such difficulties have been largely studied and reported in scientific work looking at the design aspects (e.g. [31] and [9]), at the implication on operations (see typical image of controls customization where operators add beer labels on top of control levers p. 95 [27] from [32]) and safety when incident or accident occurred ([30] p 193 on Chernobyl accident). In order to overcome such constraints, design drivers for command and control systems have been targeting at concentration[1] and integration of both displays and controls. In several domains such as control rooms and aviation, such concentration was achieved by adding computing resources for concentrating data from multiple displays into a single (or sometimes several in case of large and complex systems) display unit. In aeronautics such concentration of display is known under the notion of "glass cockpit" as computer screens were replacing previous analog displays. The benefits of such concentration had significant positive impact on operations making, for instance, large commercial aircraft operations evolve from 3 operators to only 2 in the Airbus 320 (the first commercial civil aircraft using glass cockpit technology) even though other factors such as weight were also predominant to the migration.

However, 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 thus reaching limits of user interfaces concentration. 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,

[1] By concentration we refer here to the terms coined by J. Vanderdonckt in [35]

loss of situational awareness, or skill degradation, whereas not enough automation can lead to an unmanageable, unsafe or problematic workload [29]. This is the reason why, for instance, the SESAR (Single European Sky ATM[2] Research) programme targets higher levels of automation in aviation in order to improve safety and efficiency of ATM operations.

Work on function allocation such as the ones described in [11] or [4] aim at supporting the design of automation and more precisely at identifying and assessing candidate functions to be automated. Beyond that, if the use of the system is highly dynamic i.e. evolves regularly (for instance in order to handle unexpected adverse events such as malfunctions, faults, malicious attacks …), there is a need for dedicated support to anticipating evolutions and for providing adequate solutions. This paper proposes a model-based tool-supported approach for the design and development of distributed user interfaces in the context of highly dynamic complex systems requiring repetitive and systematic activities to be allocated to the system in order to allow operators to be focussing on more analysis and decision related tasks. This approach embeds automatic generation of distributed user interfaces allowing operators to monitor the execution of semi-autonomous procedures. Next section presents with more details the context that has been introduced above. The following section presents the process associated with the approach exhibiting why there is a need of distributing the operators' user interfaces in two different parts, one being the standard command and control interface and the other one being an additional UI generated for handling a dedicated adverse event. The last section presents a case study about satellite ground segments applying step by step the approach. Finally a conclusion and directions for future work are presented.

AUTOMATION IN THE CONTEXT OF COMPLEX SYSTEMS

There are many different levels for implementing design decisions in order to include autonomous behaviors in a computing system. The first one (*static level*) consists in defining and designing the allocation at design time and to design and build the interactive system according to this allocation of functions. This is for instance the case in automotive industry with the ABS (anti-lock braking system). This autonomous system prevents vehicles wheel from blocking while the driver is breaking. Even though the autonomous system is triggered by the user, its behavior is "hard coded" and cannot be altered. The second one (*dynamic execution level*) consists in designing and defining flexible and redundant functions as in the aeronautics domain with the auto pilot. All the functions that are available in that autonomous system (such as climbing to a certain altitude) can also be performed manually by the pilot. The decision to allocate the execution of the function to the autonomous system remains in the hand of the user. The last level (*dynamic execution and definition level*) allows the user to define the behavior of the automation and also to decide when such autonomous behavior will be executed. Such level corresponds for instance to the definition and execution of macros in Microsoft Excel or the text styles in Microsoft Word.

The current paper addresses the last level (presented above) applied to command and control systems for satellite control rooms. Indeed, in case of malfunction the operator is required to define a procedure in charge of solving the identified problem. Such procedures are then tested and executed either in an autonomous or manual way. However, even in the case of autonomous execution some information might be required from the operator to complete the execution. Such information can be values of some parameters (presented on some display units) of the satellite or go/no go that contacted experts in the domain of the failure (e.g. engines, electricity …) have provided to the operator. One of the issues related to that problem is that the information required from the operator can be distributed amongst many displays making this activity cumbersome, time consuming or even error-prone. The objective of this research work is to exploit the content of the procedure defined by the operator to generate and additional user interface dedicated to the management of the procedure. This user interface gathers all the information that has to be checked and provided by the operator throughout the execution of the procedure. How such user interfaces can be generated from the definition of the procedure is presented in details in the following section. It is important to note that the point is not here to modify the existing user interface of the application but to generate an additional, contextual user interface. This prevents difficulties that may occur and which are known under the term "automation surprises" [28] if the routine interface was unpredictably altered by the generation process. Indeed, currently the new interface generated can be simply ignored, at no cost, by the operators.

USER INTERFACE GENERATION FOR DYNAMIC PARTLY AUTOMATED SYSTEM

As presented in the previous section, in the area of complex command and control systems, some of the user tasks and activities cannot be identified beforehand i.e. at design time. In addition to that issue, these tasks can be complex and/or inadequate for a human being (requiring for instance, management of a large amount of information, execution of multiple commands under strong temporal constraints, …). Such tasks are thus good candidates for delegation to an autonomous sub-system. In order to address those issues there is a need to provide operators with meta-level systems able to combine multiple commands and to delegate their execution to an

[2] Air Traffic Management

autonomous agent. The design of this part of the partly-autonomous command and control system requires the same level of reliability and usability as the rest of the application. While the reliability aspects of user interfaces can be addressed using standard dependability and fault-tolerance techniques such as the COM/MON architecture proposed by [14] and applied/extended to user interfaces in interactive cockpits [33], the usability aspects have to be addressed according to the work done in the area of automatic generation of User Interfaces as described in [34] or more recently in [26].

Several model-based approaches and toolkits aim at designing and implementing Distributed User Interfaces (DUIs) reconfigurable at runtime. Fröberg et al [10] present a framework called Marve in order to support graphical components reallocation across platform. Their work particularly focuses on event communication structure management. Melchior et al. [21] introduce a toolkit to deploy DUIs and then a framework based on state transition diagrams to represent distribution states of a DUI [22]. Kjeldsen et al. [13] also present a system architecture for widget interaction reconfiguration on planar surfaces. Another set of contributions dealing with dynamic reconfiguration of distributed user interfaces layout are based on the CAMELEON framework [5]. Manca and Paterno present a dialog model description language which aims at supporting dynamic distribution of user interfaces elements across various devices [15]. Other contributions deal with runtime architectures. Clerckx et al. [6] propose a design process and runtime architecture supporting partial dynamic redistribution of the user interface at runtime.

These contributions do not take into account or partially (in the case of state transition diagrams to represent the distribution states [21]) the behavioural part of the distributed interactive applications. This is a critical aspect when dealing with command and control of safety critical systems which might lead to deadlocks. We previously addressed that aspect by proposing fault-tolerant architectures dedicated to the dynamic reconfiguration of user interfaces in the context of cockpits of large civil aircrafts. This reconfiguration supports distribution as well as relocation of user interfaces of critical applications to other displays unit when the default one is faulty [24] and [23].

AN AUTOMATED DESIGN PROCESS FOR GENERATING INTERFACES FOR PARTLY AUTOMATED SYSTEMS

Generation of user interfaces can be envisioned if behavioural description of the automation is available and if a generic mechanism for distribution is available. However, such generation of the user interface must not have a negative impact on monitoring activities, so distribution to another display and/or to another window is required. This distribution allows decoupling the introduction of new interfaces (generated) from the set of existing ones. The design process presented in this section aims at guaranteeing the continuity of operation so that the predefined set of interfaces for monitoring and control is not altered by the generated ones.

Overview of the process

Figure 1 presents the generic process involving dynamic generation of part of the User Interface. That Figure is split in three parts.

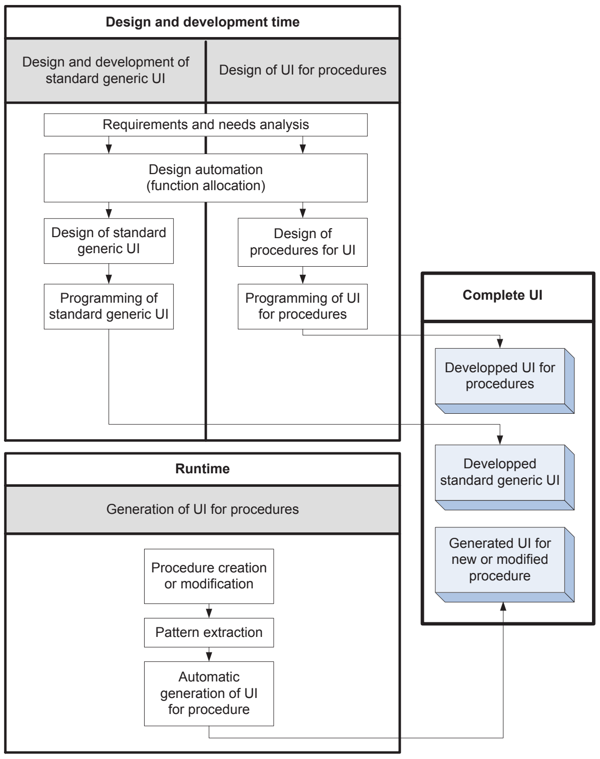

Design and development time
Design and development of standard generic UI
Design of UI for procedures
Requirements and needs analysis
Design automation (function allocation)
Design of standard generic UI
Design of procedures for UI
Programming of standard generic UI
Programming of UI for procedures
Complete UI
Developped UI for procedures
Developped standard generic UI
Generated UI for new or modified procedure
Runtime
Generation of UI for procedures
Procedure creation or modification
Pattern extraction
Automatic generation of UI for procedure

Figure 1. General overview of the approach

The first part (called Design and development time) on top corresponds to the design and development of the User Interface that is done following a classical user-centered development process. The only difference is located in the phase called (Design Automation (function allocation as defined in [4])) dedicated to the attribution of functions either to the partly-autonomous system or to the operator. Of course the description of the process remains on purpose abstract not even showing the iterations as we only highlight here the main principles. The interested reader can find a more complete and precise description of such a user-centered design process in [20]. This part is split into two threads of developments represented by the two swim lines. The right-hand side corresponds to the standard development aiming at producing a usable user interface.

The underlying concept behind this process is that there are two types of user interfaces that will be used by the operator. A generic user interface allowing the operator to perform the main tasks assigned to him/her and a set of specific user interfaces aiming at supporting specific activities defined by procedures. The generic user interface corresponds to the UI of the command and control system allowing managing the entire system while the specific UI are dedicated to procedure (that might have been defined after the UI of the command and control system has been finalized). This process is rather generic in critical systems where modification of the command and control systems might involve time and resource consuming activities such as certification by external authorities.

The other two boxes in Figure 1correspond to the design and development of the specific user interfaces dedicated to the management of specific procedures. The one on the right-hand side corresponds to procedures that have been identified during the design phases of the command and control system and follow the standard user-centered design process. The one at the bottom of Figure 1corresponds to the generation of a user interface while the command and control system is in operation. Indeed, in many cases e.g. change is usage processes or handling of unexpected adverse events not envisioned during the design phases of the command and control system. The resulting user interface of the command and control system is thus the sum of these 3 interfaces. It is important to note that the generated part does not replace the existing one but is proposed as a kind of contextual help to the operators.

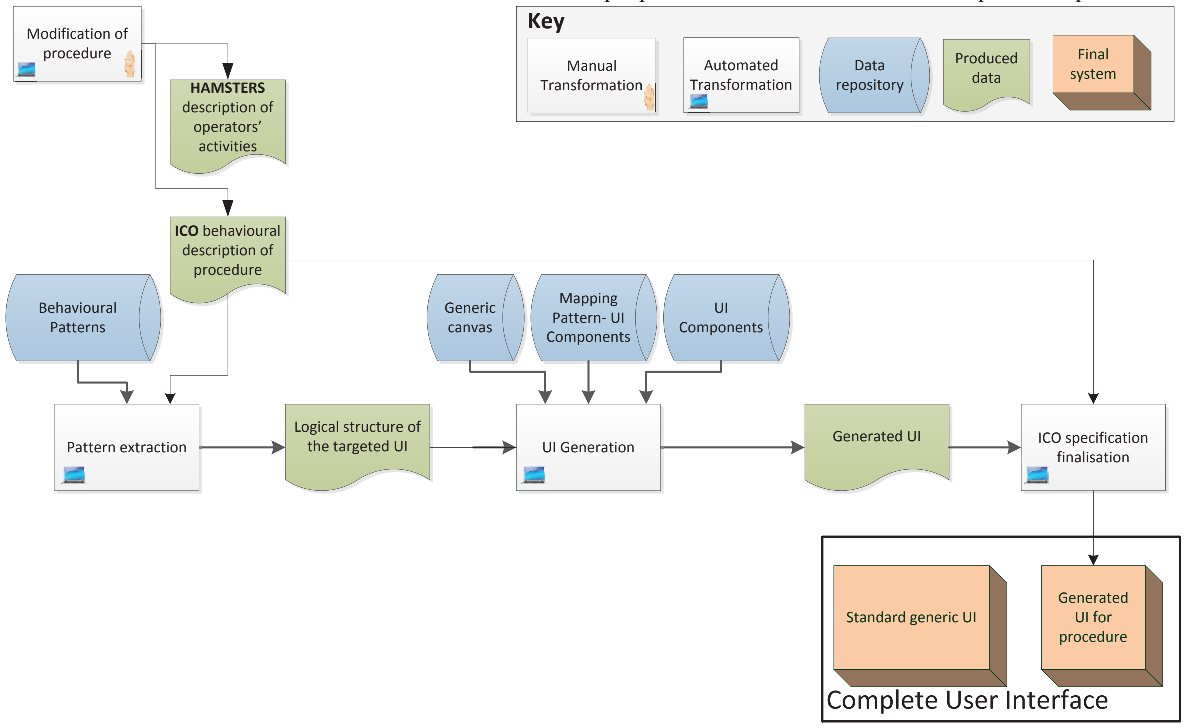

Key
Manual Transformation
Automated Transformation
Data repository
Produced data
Final system
Modification of procedure
HAMSTERS description of operators' activities
ICO behavioural description of procedure
Behavioural Patterns
Generic canvas
Mapping Pattern- UI Components
UI Components
Pattern extraction
Logical structure of the targeted UI
UI Generation
Generated UI
ICO specification finalisation
Standard generic UI
Generated UI for procedure
Complete User Interface

Figure 2. Generic generation process for the user interface of procedures

Distribution and generation

Figure 2 refines the user interface generation process presented at the bottom of Figure 1. It starts with a manual activity carried out by the operator consisting at modifying an existing (or potentially creating a new one).

- To describe the procedure (as explained with more details in the case study section) operators are provided with behavioral description languages such as YAWL [12]. Our process is based on another language called ICOs (Interactive Cooperative Objects) [25] which combines Petri nets and Object-Oriented constructs allowing manipulating values within the Petri net-based behavioral description. Beyond that, activation and rendering functions in ICO make it possible to connect this behavioral description to the graphical user interface it describes. This activity is represented as a manual and automated process as it is performed using dedicated editing tool. The ICO description of the procedure provides the grounding of the behavioral part of the user interface that will be generated.
- To describe the operators' activities that cannot be inserted in ICO models, HAMSTERS (Human-centred Assessment and Modelling to Support Task Engineering for Resilient Systems) notation is used. HAMSTERS is a task modelling notation designed for representing the decomposition of human goals into activities (perceptive, cognitive, motor, interactive…).
- The ICO procedure is then automatically analyzed using a Petri net pattern detector based on a collection of patterns descriptions. These patterns correspond to the basic bricks that constitute the procedure behavior and depend on the application it is related to. The

product of this pattern extraction is a logical structure of the targeted application as a collection of instantiated patterns (an instantiated pattern contains attributes that directly relate it to the part of the ICO description it corresponds to). As within our generation process this description is only transient, we do not handle it as a model per se, even if it would be possible.

- For each of these instantiated patterns, the UI generation phase associates a concrete component using a predefined mapping and these components are then composed within a generic graphical canvas, creating a default layout of these components. The production of this phase is a model that does not describe the behaviour of the generated application (the behaviour being provided by the ICO model in the next step). This is not presented on Figure 2 but the components, the generic canvas and the produced application are customizable, allowing a fine tuning of the produced user interface. This would be needed for instance when maintenance is performed of the application thus going back to the design process.
- Lastly, the generated model and the ICO procedure are put together to provide the final interactive user interface (using the activation function and the rendering function of ICO introduced above).

This generation process is instantiated and illustrated on a case study in the following section.

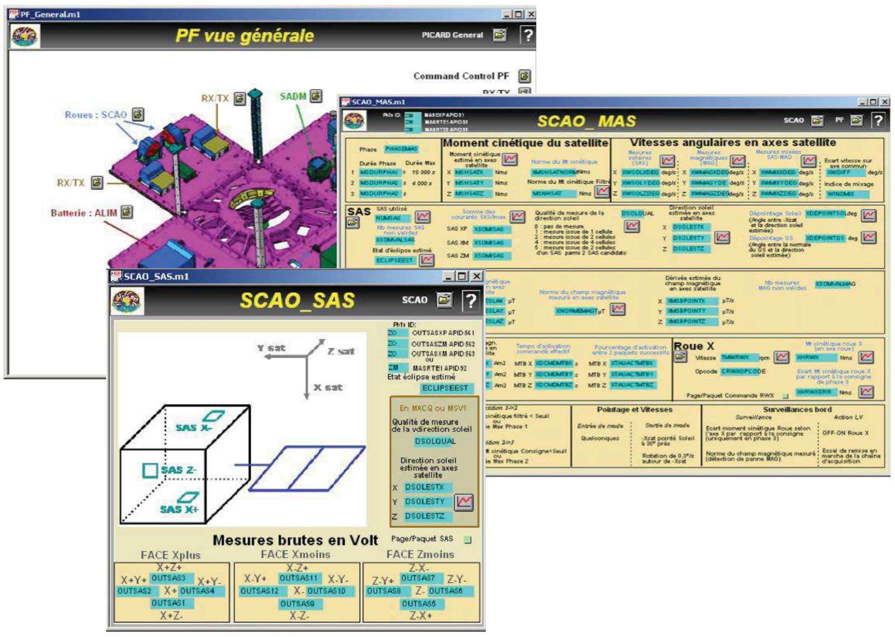


Figure 3. Examples of textual and graphical synoptics

Relationship with Previous Work

As presented above this work build on top of previous work we have done on the formal description and prototyping of user interfaces. While that previous work was focussing on supporting developers in a) identifying users activities and goals (using the notation HAMSTERS [17]) b) describing in a complete and unambiguous way both the interface and the associated interaction techniques using the ICO formal description technique [25] c) a set of case tools called CIRCUS integrating HAMSTERS case tool and ICO case tool called PetShop [1].

HAMSTERS[3] is a tool-supported graphical task-modeling notation aiming at representing human activities in a hierarchical and ordered way. Goals can be decomposed into sub-goals, which can in turn be decomposed into activities, and the output of this decomposition is a graphical tree of nodes. Nodes can be tasks or temporal operators.

The ICO formalism is a formal description technique dedicated to the specification of interactive systems [25]. It uses concepts borrowed from the object-oriented approach (dynamic instantiation, classification, encapsulation, inheritance, client/server relationship) to describe the structural or static aspects of systems, and uses high-level Petri nets to describe their dynamic behavioral aspects.

As this paper only focusses on the process and the benefits of generating specific user interface while the system in under use, next section will not present detailed models of the case study.

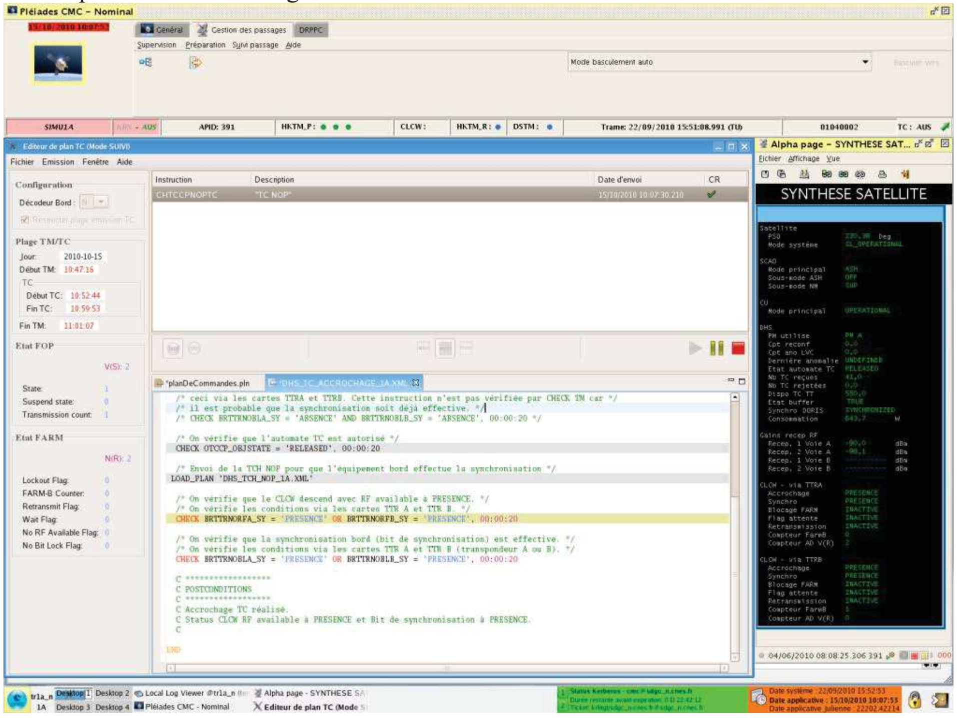
[3]http://www.irit.fr/recherches/ICS/softwares/hamsters/index.html

CASE STUDY

The example presented in this section belongs to the category of complex command and control systems from the space domain. Such interactive systems are less time constrained than other ones (such as aircraft cockpits). Beyond that, such systems are less safety critical (the only possible safety issue would correspond to a spacecraft falling on earth and injuring people). However, the potential cost of a failure is far beyond the development cost of these systems making them belong to the category of critical systems. This case study aims at highlighting how to automate the distribution of interface for the operators, providing a particular focus on the design of procedures and the generation of interactive means to control the automation. These concepts as well as the development process presented above have been applied to satellite ground segment applications within the context of the ALDABRA (Architecture and Language for Dynamic And Behaviorally Rich interactive Application) Research & Technology project.

PICARD ground segment overview

The PICARD satellite mission is dedicated to solar activity observation. Operators are in charge of two main activities: observing periodically the vital parameters of the satellite and performing maintenance operations when a failure occurs. They may have to lead concurrent activities such as monitoring satellite state and parameters, detecting failures and recovering from them, preparing and following up TeleCommand plans. To support the task of failure detection and recovery, the Operation Ground Systems is made up of two relatively unconnected components. Amongst the interactive systems used within the control room of PICARD, synoptic (see **Figure 3**) represent an important support to the operators' activities. Synoptic gather a set of parameters to propose a general overview of them, these parameters being used by the operators to monitor the state of the satellite. The PICARD operation control centre uses more than fifty synoptic containing around 10 000 parameters (such as battery status, communication link status…), and the number of procedures for possible maintenance operations goes beyond one hundred. As illustrated in **Figure 3**, synoptic may contain graphical representation of parameters, but most of them represent parameters as text (such as the central part of **Figure 3**).



```
/* ceci via les cartes TTRA et TTRB. Cette instruction n'est pas vérifiée par CHECK TM car */
/* il est probable que la synchronisation soit déjà effective. */
/* CHECK BRTTRNOBLA_SY = 'ABSENCE' AND BRTTRNOBLB_SY = 'ABSENCE', 00:00:20 */

/* On vérifie que l'automate TC est autorisé */
CHECK OTCCP_OBJSTATE = 'RELEASED', 00:00:20

/* Envoi de la TCH NOP pour que l'équipement bord effectue la synchronisation */
LOAD_PLAN 'DHS_TCH_NOP_1A.XML'

/* On vérifie que le CLCW descend avec RF available à PRESENCE. */
/* On vérifie les conditions via les cartes TTR A et TTR B. */
CHECK BRTTRNORFA_SY = 'PRESENCE' OR BRTTRNORFB_SY = 'PRESENCE', 00:00:20

/* On vérifie que la synchronisation bord (bit de synchronisation) est effective. */
/* On vérifie les conditions via les cartes TTR A et TTR B (transpondeur A ou B). */
CHECK BRTTRNOBLA_SY = 'PRESENCE' OR BRTTRNOBLB_SY = 'PRESENCE', 00:00:20

C ******************
C POSTCONDITIONS
C ******************
C Accrochage TC réalisé.
C Status CLCW RF available à PRESENCE et Bit de synchronisation à PRESENCE.
C

END
```

Figure 4. Procedure manager

Another important part of the ground segment system is the procedure manager. It aims at triggering TeleCommands, i.e. uploading commands onto the board system in order to change its current configuration and makes the parameters evolve (see Figure 4).

When operating a satellite (for instance when executing a particular procedure), such a quantity of screens and density of information makes it difficult for the operators to find a particular parameter navigating amongst the synoptic. This activity may be critical when the operator tries to solve a satellite failure, where he/she has to precisely analyse the relevant parameters. The complexity of a satellite makes it difficult to design a dedicated synoptic for each kind of failure, so that when an unexpected event occurs, dedicated procedures must be redesigned, but not the interactive system itself which remains the same (and is thus design as generic as possible).

Operational procedures as partly automated systems
Satellites and spacecraft are monitored and controlled via ground segment applications in control centres with which satellite operators implement operational procedures. A procedure contains instructions such as sending teleCommands (TC), checking teleMetry (TM), waiting, providing required values for parameters, etc. The definition of operational procedures may be found in the ECSS-E-70-32A standard [8] and defines the elements that an operational procedure must contain (declaration of the local events raised within the procedure, preconditions, instructions…). Procedures are the main mechanism used in control rooms to manage the spacecraft during both test and operations phases.

Software environment and modelling tools associated to the generation and distribution process
The targeted platform (due to the project requirements) is Java and more specifically the Java technology called JavaFX (http://javafx.com) which allows the description of the graphical part of an interactive application with an XML file (called FXML) and which allows customisation of the graphical rendering using CSS styling (http://www.w3.org/Style/CSS/).

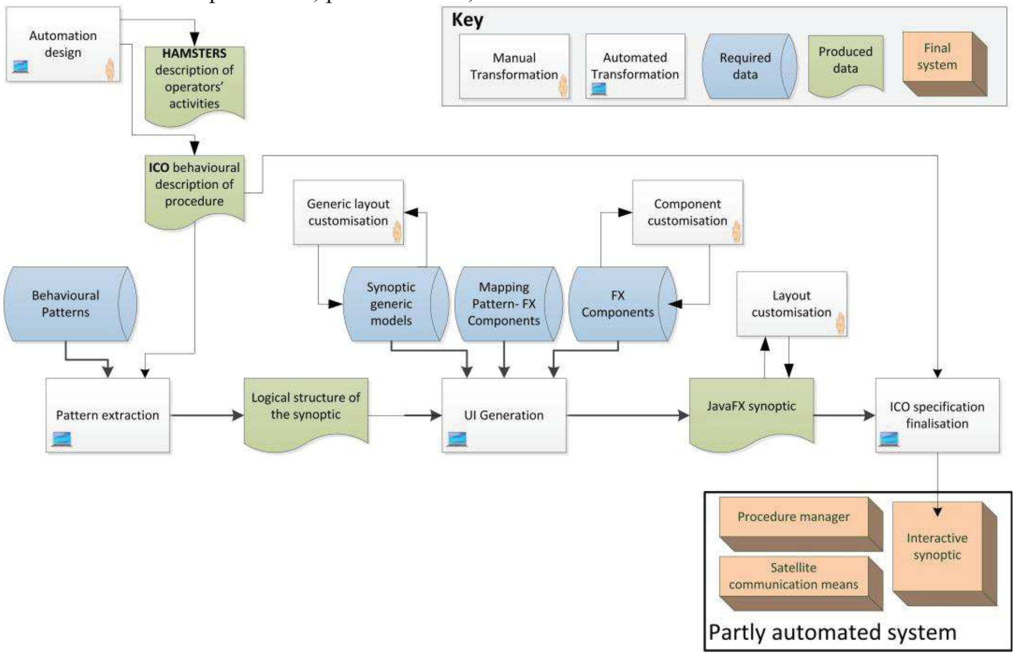


Figure 5. Generation process for interactive synoptics

HAMSTERS and ICO notations, presented in previous section, have associated CASE (Computer Aided Software Environment) tools. HAMSTERS associated software tool (also called HAMSTERS) enables to edit task models and simulate their execution. ICO (Interactive Cooperative Object) is Petri nets based and associated to a supporting tool, Petshop. It enables to edit application behavioural models and to connect them to the presentation part of the user interface (graphical widgets and frames for example). It also enables to execute the application with the underlying behavioural models. Additionally, HAMSTERS task models and Petshop system models can be connected at edition time as well as at runtime in order to ensure consistency between operator tasks and system behaviour [1, 18, 19]. This synergistic use of the two tool-supported notations also provide support for assessment of function allocation between operator and system [16].

Application of the process to PICARD ground segment applications (synoptic and procedure manager)
The main idea we illustrate with this case study is how to take benefits from the model-based generation process to support the generation of customizable interactive synoptic, and to associate them to the original interfaces (synoptic and procedure manager) that are required to support most of the activities of the operators. The generic distribution process (Figure 2) has been instantiated (Figure 5) to reflect the use of our targeted platform and modelling tools:

- The starting point of the process (top-left part of Figure 5) is the original operational procedure from which we manually produce an ICO model (and a Hamsters model that is not represented here due to space constraints).
- The ICO procedure represents the behaviour of the being generated interactive synoptic and the

modifications performed on it introduces iterations in the generation process.

- The ICO procedure is automatically analysed with a Petri net pattern detector (bottom-left part of Figure 5), associated to a collection of patterns descriptions, which embed algorithms to detect the basic bricks that constitute a procedure such as parameter update, checking of these parameters, messages and choices proposed to operators. The result of this pattern extraction is a logical structure of the synoptic in form of a list of instantiated patterns (with the list of monitored parameters and a list of elements of the control flow of the procedure).
- A JavaFX component is then associated to each of this instantiated patterns, using a predefined mapping. These components are then integrated within a generic synoptic canvas, producing a JavaFX application (with no behaviour, the behaviour being provided by the ICO model in the next step). The customisation of the JavaFX components, generic canvas and produced JavaFX application is additionally supported by the use of CSS styling to precisely adjust graphical attributes of the generated synoptic.
- Lastly, the JavaFX synoptic and the ICO procedure are put together to provide the final interactive synoptic.

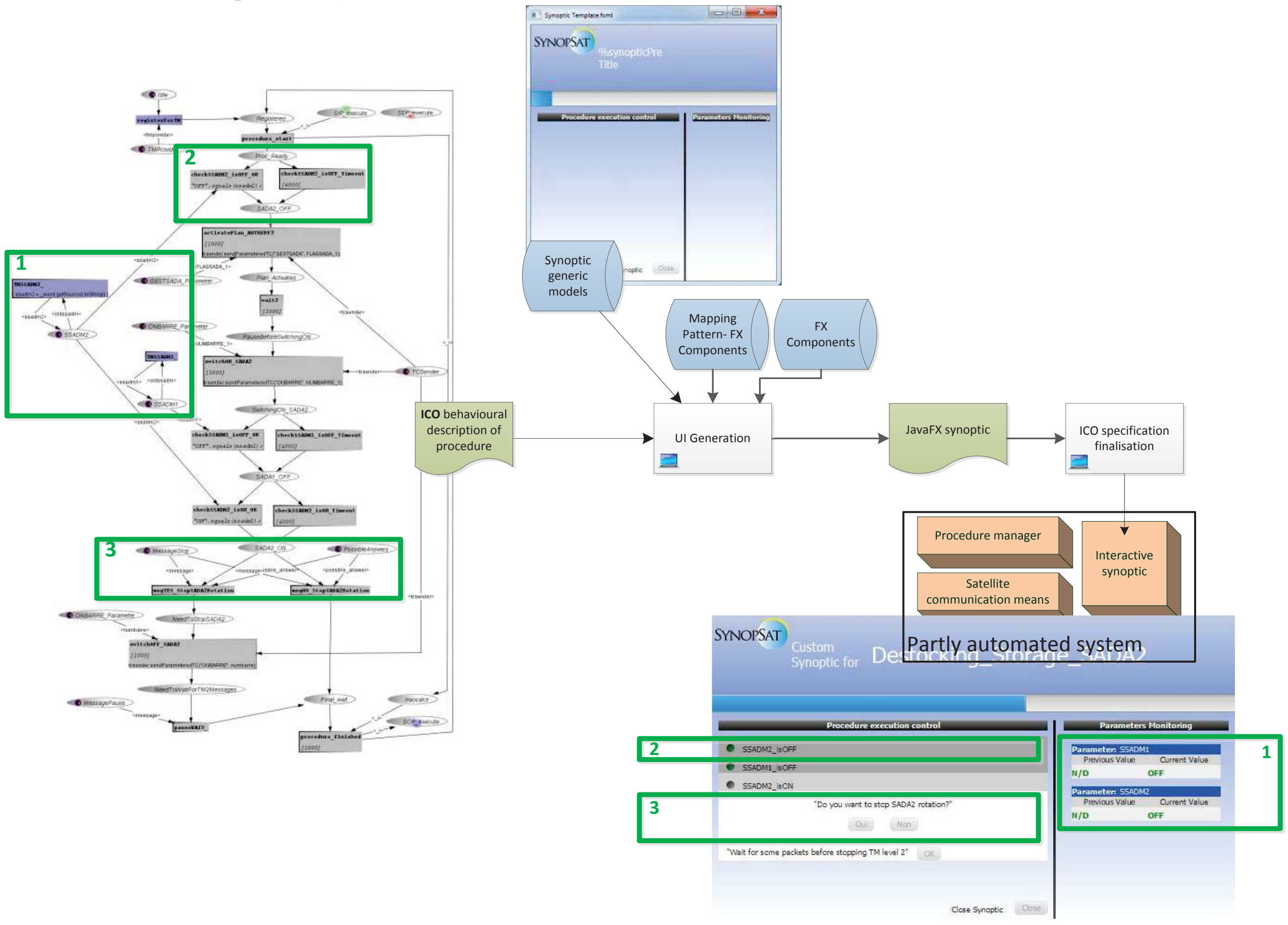

Synoptic generic models
Mapping Pattern- FX Components
FX Components
ICO behavioural description of procedure
UI Generation
JavaFX synoptic
ICO specification finalisation
Procedure manager
Satellite communication means
Interactive synoptic
Partly automated system
SYNOPSAT
Custom Synoptic for Destocking_Storage_SADA2
Procedure execution control
Parameters Monitoring
SSADM2_isOFF
SSADM1_isOFF
SSADM2_isON
"Do you want to stop SADA2 rotation?"
Oui
Non
"Wait for some packets before stopping TM level 2"
OK
Parameter: SSADM1
Previous Value
Current Value
N/D
OFF
Parameter: SSADM2
Close Synoptic
Close
1
2
3

Figure 6. Models and interactive synoptic produced during the generation process

Examples of the models and interactive synoptic produced during this generation process are presented in Figure 6:

- The left part is an excerpt of the ICO model of the corresponding procedure where two parts are highlighted, corresponding to two behavioural patterns corresponding to the two parts on the right side of the figure.
- The centre part represents the generic graphical canvas.
- The bottom-right part is the resulting interactive synoptic.

CONCLUSION

This article has presented how model-based approaches can be used for the automated generation of contextual user interfaces and how they can provide operators of ground segments with focus and context information. This approach exploits a formal behavioural description technique (the ICO notation [25]) for the description of both the operational procedures and thus the behaviour of the generated user interface. The graphical presentation is produced using an XML dialect called FXML which belongs to the JavaFX technology. This contribution presents a unique case study where the generation of user interfaces provides important benefits for operators of critical interactive systems. Furthermore, the distribution of generated user interface across another display guarantees segregation with the standard command and control system thus preventing possible fault propagation to the ground segment.

The current work corresponds to the final contribution of the research project ALDABRA and is under consideration for inclusion in the next generation of ground segment operations. While informal testing with ground segment operators has received very positive feedback, the critical system nature of the application domain requires adoption by regulatory authorities prior to development (by certified companies) and deployment in operational satellite ground segment. Such work is being undertaken and lead by CNES via ISIS (Initiative for Space Innovative Standards) targeting at standard, generic and innovative ground segments (http://www.iafastro.net/iac/archive/browse/IAC-09/B4/7/4801/). This work is part of a more ambitious research programme aiming at defining processes, methods and tools for the design and development of safety critical interactive systems. While function allocation is critical for most (partly-) autonomous systems, the current paper only referred to a context of automation where allocation is previously defined and does not evolve. Future work intends to extend previous work on automation design [16] and aims at exploiting the tasks models to identify potential migrations and to assess the impact of such migrations on operations' performance.

ACKNOWLEDGMENTS

This work was partly sponsored by CNES (French National Space Studies Center) R&T Tortuga R-S08/BS-0003-029 and Aldabra ETS-CT-R&T_TTGA-196-CN as well as EUROCONTROL research network HALA! on Higher Automation Levels in Aviation and WPE project SPAD (System Performance under Automation Degradation).

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. ACM SIGCHI conference Engineering Interactive Computing Systems (EICS 2010), ACM SIGCHI, p. 143-152, 2010.

2.Barboni, E, Martinie C., Navarre, D, Palanque, P, Winckler, M. Bridging the Gap between a Behavioural Formal Description Technique and User Interface Description Language: Enhancing ICO with a Graphical User Interface Markup Language. Journal of Science of Computer Programming Vol. 78, 2013.

3.Bastide R., Sy O. & Palanque P. A formal notation and tool for the engineering of CORBA systems. Concurrency - Practice and Experience 12(14): 1379-1403 (2000)

4.Boy G. Cognitive Function Analysis for Human-Centered Automation of Safety-Critical Systems. Proc. of ACM SIGCHI conference on Human Factors for Computing Systems 1998: 265-272

5.Calvary, G., Coutaz, J., Thévenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A Unifying Reference Framework for multi-target user interfaces. Interacting with computers, 15(3), 1-1.

6.Clerkx, T., Vandervelpen, C., Coninx, K. Task-based design and runtime support for multimodal user interface distribution. In Proceedings of Engineering Interactive Systems 2007, EHCI-HCSE-DSVIS (2007).

7.European Cooperation for Space Standardization, Space Engineering, Ground Systems and Operations, ECSS-E-70C, 31 July 2008.

8.European Cooperation for Space Standardization, Space Engineering,Test and Operations Procedure Language, ECSS-E70-32A. 2006.

9.Fang Chen, Eric H. C. Choi, Natalie Ruiz, Yu Shi, and Ronnie Taib. 2005. User interface design and evaluation for control room. In Proceedings of the 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future (OZCHI '05). Computer-Human Interaction Special Interest Group (CHISIG) of Australia, Narrabundah, Australia, Australia, 1-4.

10.Fröberg, A., Eriksson, H., Berglund, E. Developing a DUI Based Operator Control Station: A Case Study of the Marve Framework. . In J.A. Gallud et al. (eds), Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem, Human-Computer Interaction Series, pages 1-12, 2011, Springer-Verlag, 2011.

11.Harrison, M., Johnson, P., and Wright, P. (2002). Automating functions in multi-agent control systems: supporting the decision process. In Redmill, F and Anderson, T. editors, Proceedings of the Tenth safety-critical system symposium, Southampton. Springer. pp. 93-106.

12.Hofstede, A. H. M. Ter., YAWL: yet another workflow language. Information Systems. 2005, Vol. 30, pp. 245-275.

13.Kjeldsen, R., Levas, A., Pinhanez, C. Dynamically Reconfigurable Vision-Based User Interfaces. In 3rd International Conference on Vision Systems (ICVS'03). Graz, Austria. April 2003.

14.Laprie, J-C., Arlat, J., Béounes, C. & Kanoun, K., Definition and Analysis of hardware and software Fault-Tolerant Architectures, IEEE computer, vol.23, no.7, pp.39-51, 1990

15.Manca, M., Paterno, F. Extending MARIA to support Distributed User Interfaces. In J.A. Gallud et al. (eds), Distributed User Interfaces: Designing Interfaces for the

Distributed Ecosystem, Human-Computer Interaction Series, pages 1-12, 2011, Springer-Verlag, 2011.

16.Martinie C., Palanque P., Barboni E. & Ragosta M. Task-model based assessment of automation levels: Application to space ground segments. IEEE SMC conference, IEEE explore, 2011: 3267-3273

17.Martinie C., Palanque P., and Winckler M. Structuring and composition mechanisms to address scalability issues in task models. IFIP TC 13 int. conf. on Human-computer interaction - Volume Part III (INTERACT'11), Vol. Part III. Springer-Verlag, Berlin, Heidelberg, 589-609.

18.Martinie, C, Palanque, P, Navarre, D, Barboni, E. A Tool-Supported Training Framework for Improving Operators´ Dependability Confronted with Faults and Errors. Probabilistic Safety Assessment (PSAM11 & ESREL 2012), Helsinki, Finland, June 25-29 2012, Taylor & Francis Group.

19.Martinie, C, Palanque, P, Navarre, D, Winckler, M., Poupart. Model-based training: an approach supporting operability of critical interactive systems. ACM SIGCHI conference Engineering Interactive Computing Systems (EICS 2011), pp.53-62, ACM press.

20.Martinie C., Palanque P., Navarre D. and Barboni E. A Development Process for Usable Large Scale Interactive Critical Systems: Application to Satellite Ground Segments. 4th IFIP International Conference on Human-Centred Software Engineering (HCSE 2012), Springer Verlag, LNCS

21.Melchior, J., Grolaux, D., Vanderdonckt, J., Van Roy, P. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In Proceedings of the 1st ACM SIGCHI symposium on Engineering Interactive Computing Systems (EICS 2009), pp. 69-78, ACM New York, 2009.

22.Melchior, J., Vanderdonckt, J., Van Roy, P. A Model-Based Approach for Distributed User Interfaces. ACM SIGCHI symp. on Engineering Interactive Computing Systems (EICS 2011), p. 11-20, ACM New York, 2011.

23.Navarre, D., Palanque, P., Basnyat, S., (2008) Usability Service Continuation through Reconfiguration of Input and Output Devices in Safety Critical Interactive Systems. 27th Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP 2008), LNCS 5219, pp. 373–386, 2008. © Springer-Verlag Berlin Heidelberg 2008.

24.Navarre, D., Palanque, P., Ladry, J.F., Basnyat, S. Architecture and a Formal Description Technique for the Design and Implementation of Reconfigurable User Interfaces. In Proceedings of the 15th International Workshop on Interactive Systems. Design, Specification, and Verification (DSV-IS 2008), LNCS 5136, pages 208-224, Springer-Verlag Berlin Heidelberg 2008.

25.Navarre D., Palanque P., Ladry J-F & Barboni E: ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interact. 16(4): (2009)

26.Nichols J., Duen Horng Chau, and Brad A. Myers. 2007. Demonstrating the viability of automatically generated user interfaces. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '07). ACM, New York, NY, USA, 1283-1292.

27.Norman D. (1998). The design of everyday things. MIT press 1998.

28.Palmer, E. "Oops, it didn't arm." - A Case Study of Two Automation Surprises. 8th International Symposium on Aviation Psychology, Ohio State University, (1995).

29.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.

30.Reason, J. (1990). Human Error, Cambridge University Press.

31.Rohn J. Petersen, William W. Banks, and David I. Gertman. 1982. Performance-based evaluation of graphic displays for nuclear power plant control rooms. In Proceedings of the 1982 conference on Human factors in computing systems (CHI '82). ACM, New York, NY, USA, 182-189.

32.Seminara, J. L., Gonzalez, W. R., & Parsons, S. 0. (1977). Human factor review of nuclear power plant control room design (NP-309). Palo Alto, CA: Electric Power Research Institute.

33.Tankeu-Choitat A., Fabre J-C., Palanque P., Navarre D., Deleris Y., Fayolas C. Self-Checking Components for Dependable Interactive Cockpits using Formal Description Techniques. 17th Pacific Rim Dependable Computing Conference (PRDC 2011), Pasadena, US, IEEE, 12-15th December 2011.

34.Vanderdonckt Jean, Automatic Generation of a User Interface for Highly Interactive Business-Oriented Applications, San Francisco: Morgan Kaufmann, 1998, p. 516-520.

35.Vanderdonckt, J. Distributed User Interfaces: How to Distribute User Interface Elements across Users, Platforms, and Environments. In Proceedings of XIth Congreso Internacional de Interacción Persona-Ordenador Interacción'2010, AIPO, Valencia, 2010, pp. 3-14, Keynote address.