A Development Process for Usable Large Scale Interactive Critical Systems: Application to Satellite Ground Segments

Célia Martinie[1], Philippe Palanque[1], David Navarre[1], Eric Barboni[1]
[1] ICS-IRIT, University of Toulouse 3, 118 route de Narbonne,
31062 Toulouse Cedex 9, France
{martinie, palanque, navarre, barboni}@irit.fr

Abstract. While a significant effort is being undertaken by the Human-Computer Interaction community in order to extend current knowledge about how users interact with computing devices and how to design and evaluate new interaction techniques, very little has been done to improve the reliability of software offering such interaction techniques. However, malfunctions and failures occur in interactive systems leading to incidents or accidents that, in aviation for instance, are [22] 80% of the time attributed to human error demonstrating the inadequacy between the system and its operators. As an error may have a huge impact on human life, strong requirements are usually set both on the final system and on the development process itself. Interactive safety-critical systems have to be designed taking into account on an equal basis several properties including usability, reliability and operability while their associated design process is required to handle issues such as scalability, verification, testing and traceability. However, software development solutions in the area of critical systems are not adequate leading to defects especially when the interactive aspects are considered. Additionally, the training program development is always designed independently from the system development leading to operators trained with inadequate material. In this paper we propose a new iterative design process embedding multiple design and modeling techniques (both formal and informal) advocated by HCI and dependable computing domains. These techniques have been adapted and tuned for interactive systems and are used in a synergistic way in order to support the integration of factors such as usability, dependability and operability and at the same time in order to deal with scalability, verification and traceability.

Keywords: Software engineering, formal methods, task modeling, safety management, model-based design, training.

1 Introduction

Currently, most of the contributions in HCI are related to innovative interaction techniques, reports on User-Centred Design products and systems, prototyping techniques, usability evaluation and user experience. Most of the proposed techniques and methods are usually targeting mass market applications. When dealing with usability they focus on ensuring that the user will accomplish her/his goals in an

efficient way, without error and with a maximum satisfaction while when user experience is concerned, they focus on flow, emotions, ... In safety-critical contexts, effectiveness and efficiency to accomplish user tasks are also very important but there are additional constraints to be accounted for on the system being used such as reliability and operability. Additionally, beyond these constraints on the final products, constraints also apply to the development process itself. Requirements for designing and developing safety-critical systems (as defined in standard such as DO-178B [16] or ESARR 6 [15] widely used in aeronautics) aim at ensuring that the various steps of the process are traceable (and have been traced) and that risk analysis has been carried out carefully demonstrating that risk of failure or malfunctions remains at a tolerable level. On the human side, users of such systems are trained to use the system, being required to follow operational procedures and behaving in an appropriate way according to both safety regulations and mission constraints.

This paper proposes a new development process able to address interactive systems issues, safety critical concerns in system development as well as human factors issues. To this end, we integrate in this process, methods, techniques and tools which aim at taking into account usability, reliability and operability. This process is centred on models which are the only way to deal with large systems and complex operator activities. In previous work we have a set of tool-supported notations that can be used to design reliable, usable and operable interactive critical systems [26]. This process enables seamless integration between formal behavioural models of the system components and informal behaviours expressed in standard programming languages. Beyond this computer science view it integrates formal and informal techniques (such as prototyping and usability evaluation) which are necessary when dealing with human factors aspects.

The first section of the paper highlights the key issues that need to be addressed to design and develop interactive safety-critical systems. This section highlights expected properties of the systems to be built (usability, operability, safety and reliability) as well as requirements on the process itself (support for verification, traceability, scalability and to support construction of associated material such as training programme of operators). The second section presents existing design processes that support the development of interactive system. The third section presents the new development process and explains how this process produces usable, dependable and safe interactive systems and how it meets process requirements in the area of safety critical systems. The last section exemplifies how this process has been implemented to design and develop a satellite ground segment application and presents why and how requirements identified in section 2 are met.

2. Requirements for design and development of interactive systems in safety critical contexts

We propose to analyze two kinds of concerns that are still open issues in the safety-critical domain: nonfunctional requirements for the system, but also requirements on the development process. These two types of requirements are detailed in the next paragraphs.

2.1 Requirements for the system to be produced

This sub-section details which properties have to be taken into account during the development process of a safety-critical interactive system and the reasons why.

2.1.1 Usability and operability

Safety critical systems aim at achieving safety-critical missions (e.g. controls a spatial aircraft, carry people from a destination to another, monitoring a power plant...). Humans that interact with such systems operate the system, i.e. they use it according to predetermined procedures, predetermined tasks and predetermined behaviors they have been trained to master, in contexts that have been analyzed during the system design.

Usability term usually refers to effectiveness, efficiency and satisfaction. But, designed safety-critical interactive systems are also required being operable, i.e. they are required providing all of the needed functionalities so that the user is able to control the system and to accomplish all the possible tasks she/he can be requested to perform. Additionally, designed system has also to be error-tolerant and safe. Since a few years, these properties have been tackled by consolidated cross-domain taxonomies of usability, generally from a software perspective. Operability, Error tolerance [2] and software safety [45] have been added to the usability definition.

2.1.2 Reliability

It is now widely agreed upon that in order to easier to use systems, generic and specific functionalities have to be offered to the users of such systems. However, increasing the number of functionalities by adding, for instance, undo/redo mechanisms, WYSIWYG facilities increases the likelihood of system crashes. In particular, the current reliability level of software suites proposed by main software companies clearly show that building reliable interactive software is still a challenge for software development teams. While rebooting the systems is the most reasonable action in order to recover from a software crash, this is totally unacceptable in real-time safety critical systems where people life is at stake.

2.1.3 Safety

This property aims at classifying systems which *will not endanger the human life or the environment* [46]. Current User-Centred Design approaches and model-based design methods do not explicitly account for potential erroneous human and technical behaviour. But, particular attention is paid to the design and development of a safety-critical system, especially with the transversal activity of safety management. Goal is to ensure that the risk associated with the use of the system is tolerable. And risks are usually quantified and classified to provide a common understanding of tolerance. Safety integrity levels (SIL) enable to characterize the effects of a failure condition in the system. The failure conditions are usually categorized by their effects on the system, users and environment. Table 1 presents a classification of safety integrity levels for space systems 16420-1 [20]. Another example is the classification for aircraft software DO178-B [16], which identify 5 safety integrity levels (Catastrophic, Hazardous, Major, Minor, No effect).

The safety expert has to analyse incidents and accidents that occurred on similar systems, in order to prevent accidents from re-occurring. Identification of system functions or boundaries, together with hazards and risks analysis enables to determine required safety integrity levels for each system function. Hazards and risks analysis are part of the safety assessment process and consists in examining the effect of a failure condition in the system. A safety integrity level is associated to each part of the system after that a safety analysis has been performed by an expert.

Table 1. Severity of identified hazards (ISO 16420-1)

Severity	**Consequence**
1) Catastrophic hazards	i) loss of life, life-threatening or permanently disabling injury or occupational illness, loss of an element of an interfacing manned flight system;
	ii) loss of launch site facilities or loss of system;
	iii) severe detrimental environmental effects.
2) Critical hazards	i) temporarily disabling but not life-threatening injury, or temporary occupational illness;
	ii) major damage to flight systems or loss or major damage to ground facilities;
	iii) major damage to public or private property; or
	iv) major detrimental environment effects.
3) Marginal hazards	minor injury, minor disability, minor occupational illness, or minor system or environmental effects.
4) Negligible hazards	less than minor injury, disability, occupational illness, or less than minor system or environmental damage.

2.2 Requirements for the design-and-development process

This section details the concerns taken into account for the development process of a safety-critical interactive system including support to handle scalability, verification and traceability and training.

2.2.1 Scalability

The complete specification of interactive application is now increasingly considered as a requirement in the field of software for safety critical systems due to their increasing use as the main control interface for such systems. As the user interface as a part of command and control systems may represent a huge quantity of code, User Interface Tools must provide ways to address this complexity. Support only dealing with code management is not enough and there is thus a critical need for addressing this complexity at a higher level of abstraction. This paper argues that one possible way to deal with these issues is to follow the same path as in the field of software engineering where modeling activities and model-based approaches take the lead with standards like UML. Several contributions argue for this approach [38], and especially when new interaction techniques have to be addressed during the development process (such as the so-called Post-WIMP [21] or animations [14]).

2.2.2 Verification

Verification techniques aim at providing ways for ensuring systems reliability prior to implementation. User Interface Tools that would provide such capabilities would empower developers by offering means for reasoning about their systems at a higher level of abstraction.

Formal description techniques support verification and make it possible to assess properties such as: whatever state the system is in, at least one interactive object is enabled, mutual exclusion of actions, reachability of specific states [36]. Testing activities also support verification and model-based approaches [8] featuring formal description techniques can provide support to developers in the later phases of the development process where the intrinsic nature of interactive systems makes them very hard to address properly otherwise. For instance, the event-based nature of interactive systems makes them impossible to test without tool support for the generation of the possible test cases. Work in this field is still preliminary but contributions are available for WIMP interaction techniques providing means for regression testing [29] and coverage criteria [30].

2.3.3 Traceability

Traceability of requirements throughout the whole design and development process is explicitly required for safety-critical systems, from system requirements to all source code or executable object code. DO-178B [16] requires the use of methods and techniques for systematically exploring design options and for supporting the traceability of design decisions. Similarly, ESARR [15] on Software in Air Traffic Management Systems explicitly requires traceability to be addressed in respect of all software requirements. However, such standards only define what must be done in terms of traceability but provide no information on how such goals can be reached by analysts and developers.

2.3.4 Training

Human operating a safety-critical system has to be trained and qualified [26]. Users of such systems have to also to be "adapted" (or prepared) to use the system. Indeed, training is the mean to achieve this goal and is therefore mandatory for users of critical systems. Related work presented in [1] and [43] argue that training enables:

- Ensuring that operators have reached a required level of skill and knowledge before using the system.
- Enhancing and maintaining users' performances.
- Decreasing the number of human errors while using the system.

Training programs of safety critical systems have to be designed in such a way that the future user has been evaluated as being able to operate the system she/he has be trained to use. To achieve this goal, Systematic Approaches to Training are widely used across application domains of safety-critical systems [23] [26]. Unfortunately, users of interactive critical systems have not always been trained on the system they are going to use. In best cases, they have been trained on simulators of real systems that mimic the expected system's behavior. The development process should provide a way to ensure that users are prepared to the complete system's behavior.

3 Limitations of existing design processes for interactive systems

Proposing processes for the development of software systems has been a recurring activity of researchers and practitioners in the area of software engineering. Indeed,

managing large scale software systems requires structured and systematic approaches for coping with the complexity.

3.1 Legacy software development processes

The early waterfall model proposed in the 70s [43] is made up of eight steps ranging from "system requirement" phase to "operation" phase. While such structured processes (and the following versions such as the V model from [28]) promote the construction of reliable software by building the "system right", they have also demonstrated their difficulty in building the "right system" i.e. a system corresponding to the needs of the various stakeholders especially in the context of unstable and evolving requirements. To try to address such concerns the spiral development process promoted by Boehm [9] has introduced the production of specific artifacts called prototypes in order to first identify the adequacy of the current version of the software with clients' requirements, and second provide a framework for handling explicitly iterations. It took nearly 10 years (and a significant increase in software size and complexity) to understand that such iterative processes were not delivering as expected, as demonstrated by a thorough study of more than 8000 project in 382 companies reported by the same Barry Boehm [10] in 2006. As identified in this study, the main drawback of these early software development processes (beyond the inherent difficulty of building large and complex system products) was the difficulty to identify user needs and to produce software meeting both those needs and to encompass ever evolving new technologies.

3.2 User centered software design processes

Even though it took a long time to make its way in the area of software engineering, the necessity of designing software compliant with user need and user capabilities has been recognized as critical in the area of Human-Computer Interaction much earlier. The User Centered Design approach (introduced in [33]) has promoted to place user-related consideration at the center of the development processes. Several processes have since been proposed to take into account usability while designing an interactive system. Hartson et al. [18] and Collins [12] identified mandatory steps to design usable system. Curtis & Hefley [13] first tried to match software development processes with usability engineering and techniques. Rauterberg [40] identified more precisely design steps to involve end-users in an iterative-cyclic development process. Goränsson et al. [17] proposed a design process centered on usability: "*The usability design process is a UCSD approach for developing usable interactive systems, combining usability engineering with interaction design, and emphasizing extensive active user involvement throughout the iterative process*". This work highlights that design processes for interactive systems are required to be highly iterative and to promote multiple designs through evolvable prototypes in order to accommodate requirements changes and results from usability evaluations. However, such processes have put too much emphasis on the user side forgetting the complex reality of software development.

3.3 Agile approaches to software development

Iterative or agile approaches, such as Scrum [47], advocate that requirements tuning is performed by means of rapid and systematic iterations. However, including the last version of Scrum released end of 2011 there is still no reference to end user. Validation of prototypes by clients will not provide feedback to developers about compatibility with users' tasks and activities for instance. This has been clearly stated and identified in [48] where User Centered and Agile approaches where compared and assessed. Beyond that, task/artifact lie cycle as identified in [11] adds a new dimension to user needs evolution. Indeed, as described in the studies reported in that paper, the fact of providing users with new tool (even if the tools are perfectly in line with their needs) will change the needs as the work and practice of users will evolve due to this particular new tool. This demonstrates the need to involve end users throughout the development process to test, validate the systems and redefine their needs, as promoted by several research contributions [19].

Another very different problem lays in the iterative nature of the agile and spiral processes. Indeed, (as advocated by the early development processes such as waterfall or V) without identified phases gathering in one same location all the information required, software development will be chaotic resulting in hard to manage, test and modify software that has been built adding regularly new functionalities without following a global and thorough design. While this might not be a big problem when small and rather simple applications are considered, when it comes to large scale and complex systems (as satellite ground segments) this might have significant impacts both in term of development costs and resources but also in terms of reliability and dependability. To handle such complexity model-based approaches such as UML or [37] provide abstraction and domain specific notations. However, approaches such as Scrum or the Spiral model reject the use of models due to the cost in terms of effort and time.

Table 2 presents a synthetic view on coverage of requirements by types of development processes, which have been presented in this section. It highlights the absence of appropriate development processes for interactive critical systems, as no one of them is covering all of the requirements. During, the past 20 years, various contributions issued by the authors targeted to create, evaluate and enhance scalable techniques, methods, notations and tools to support the design of interactive systems which could fulfill usability, reliability and safety properties. These contributions support the creation of a new development process which takes into account requirements for the system to be produced, but also requirements for the design and development process.

Table 2. Synthesis of coverage of requirements by types of software development processes (TIA= Taken Into Account – NA= Not Addressed)

	Requirements for the interactive critical system to be produced			**Requirements for the design and development process**			
	Usability	**Reliability**	**Safety**	**Scalability**	**Verification**	**Traceability**	**Training**
Legacy development processes	NA	TIA	TIA	TIA	TIA	TIA	NA

User Centered Design processes	TIA	NA	NA	NA	NA	NA	NA
Agile approaches	NA	NA	NA	TIA	TIA	NA	NA

Next section presents the new development process which is iterative (to support incremental developments and evolution of needs and requirements) integrating task models and end-user evaluation (to handle the always evolving users' needs and to ensure usability) while proposing extensive use of models (to ensure reliability of the software). In this next section, at each stage of the development process, one or more references are cited to indicate the technique, notation, tool and/or method used to support this development process.

4. A development process for safety critical interactive systems

The proposed process takes into account previously presented system properties and previously presented development process constraints. It leverages informal HCI techniques (including mock-ups, low-fidelity prototyping, field studies…) and formal HCI techniques (including formal description techniques, formal analysis, performance evaluation…) to address usability, reliability and operability properties that generally not targeted simultaneously.

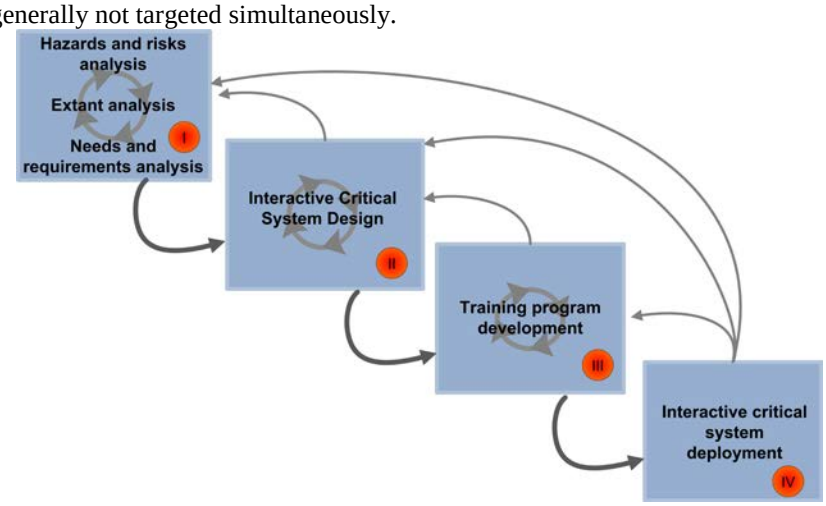


Fig. 1. Abstract view of the development process

Fig. 1 presents an abstract view of an iterative development process for critical interactive applications. As stated in the introduction the models makes explicit the design and construction of a training program and the required iterative aspect (doted arrows) for addressing users' needs and required changes following a usability evaluation. It also exhibits:

- The required activity of traceability of choices and decisions throughout this development process (by the large arrow on the right-hand side of the diagram).
- The required activity of safety management (by the large arrow on the left-hand side of the diagram), which is also a transversal activity that starts in the early stage of the design project with hazards and risks analysis activities and that will end when the system will not be in use anymore.

The first phase (disc I in Fig. 1) includes safety analysis activities such as incident and accidents analysis, existing systems analysis (or extant analysis), hazards and risks analysis. At the end of this first phase, needs and requirements for the system have been set, including safety integrity levels for each part of the system. The next step of interactive critical system design (disc II in Fig. 1) is triggered and will issue a very high-fidelity prototype as well as several types of models and descriptions for the next phases of training program development (disc III in Fig. 1) and system deployment (disc IV in Fig. 1).

4.1 Interactive Critical System Design

Fig. 2 presents a more detailed view of this development process making explicit the three specific sub-phases of the interactive critical system design process:
- Task analysis and modeling phase (under discs 3 and 7 in Fig. 2, detailed in §4.2)
- Low-fidelity prototyping iterative phase (loop represented by discs 2, 3 4 and 5 in Fig. 2, detailed in §4.3)
- Very-high fidelity prototyping iterative phase (loop represented by discs 6, 7, 8, 9 and 10 in Fig. 2, detailed in §4.4)

4.2 Task analysis and task modeling phase

Task analysis and task modeling (discs 3 and 8 in Fig. 2) aim at understanding and describing user activities. This key step in the process enables to ensure that:
- The system is providing the complete set of needed functionality to support user activities (related to effectiveness property of the usability factor).
- The user will be able to accomplish his/her goals in an acceptable timeframe while using the system (related to the efficiency property of the usability factor).

Task analysis and modeling activities (discs 3 and 8 in Fig. 2) may have been started in earlier phases (as described in paragraph “Needs and requirements analysis”), however, this activities are central to the design phase of the proposed process. It highly supports the design of a usable interactive system as it enables to identify precisely goals, tasks and activities that have to be led by the operator. Task models bring additional advantages to task analysis: the structuration of the gathered information about operators’ activities and the possibility to use software tools to compute, analyze and simulate these models. When supported by a task modeling notation and tool featuring human tasks refinement (cognitive, motor, perceptive tasks) and complex activities edition and simulation, this step enables qualitative analysis of user or operator tasks (disc 4 and 9 in Fig. 2).

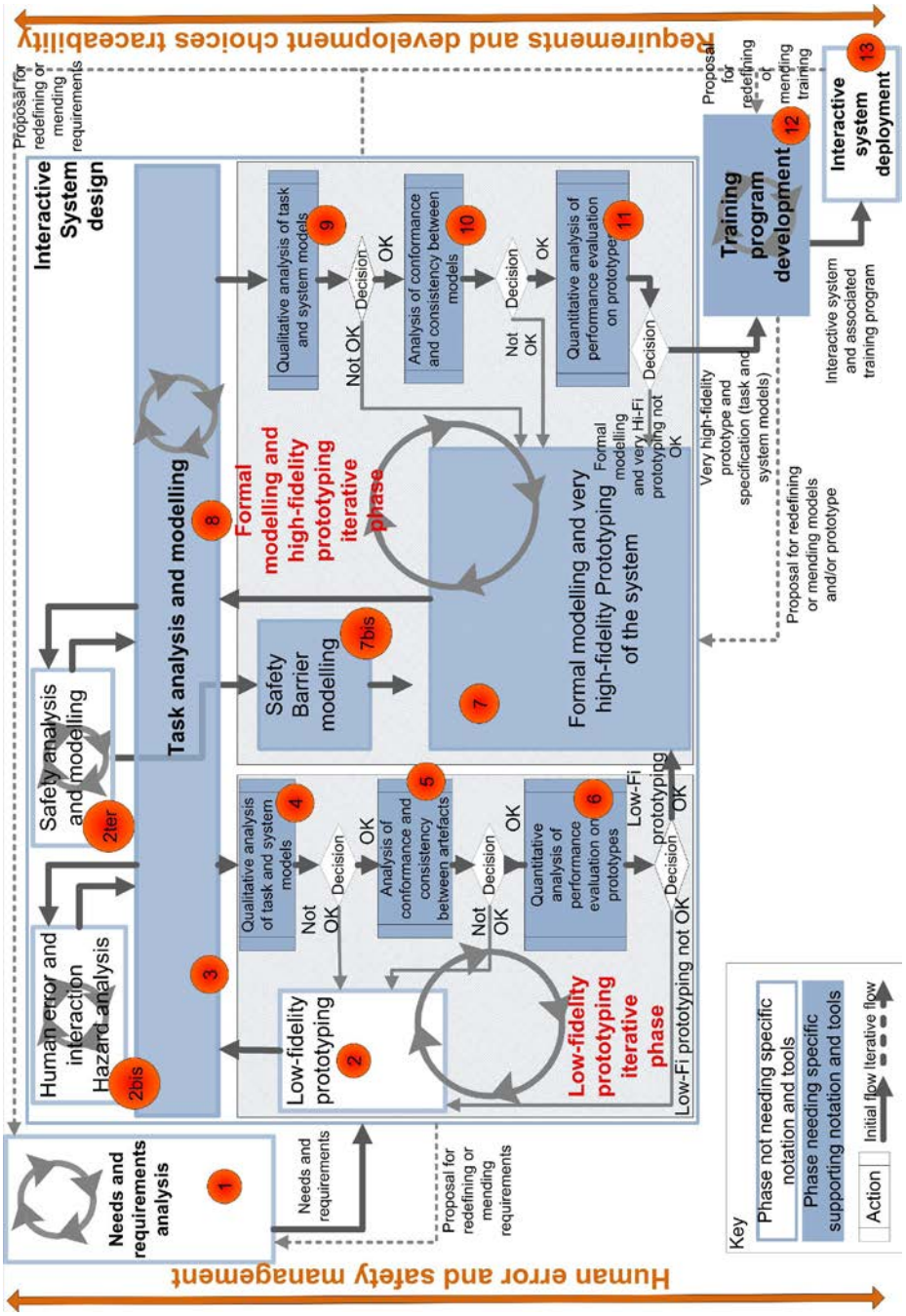


Fig. 2. Detailed view of the development process

Task analysis and modeling activities are also supports for:

- Task deviation analysis (disc 2bis in Fig. 2)

- Interactions safety analysis (disc 3ter in Fig. 2)
- Conformance analysis between prototypes and user task (disc 5 in Fig. 2)
- Conformance analysis between task models and system behavioral models by connecting the input/output event handlers to the interactive input/output user tasks (disc 10 in Fig. 2).
- Human error and task deviation analysis [35] in case of high safety integrity levels (discs 2bis and 2ter in Fig. 2).
- Quantitative analysis of user performances (disc 11 in Fig. 2).

4.3 Low-fidelity prototyping iterative phase

Low-Fi iterative phase (discs 2, 3, 4, 5 and 6 in Fig. 2) aims at preparing first versions of the interactive system and enables to evaluate first design outcomes without engaging too much human and financial resource at this stage of the process. Furthermore, it is also a first detection steps for potential safety issues.

Once low-fidelity prototypes are compliant with user tasks, the validated materials can be forwarded to the next phase.

4.4 Formal modeling, informal modeling and very high-fidelity prototyping

Very Hi-Fi prototyping and modeling phase (discs 7, 7bis, 8, 9, 10 and 11 in Fig. 2) is heavily based on two types of models: task models and system models. One of the critical aspects of having several models for the same interactive application is to support the resulting activity of ensuring conformance and compatibility of these models. We already proposed several ways of addressing such compatibility in [6] but it has been extended in order to deal also with the training and operational concerns [26]. Formal description techniques are the only means for both modeling in a precise and unambiguous way all the components of an interactive application (presentation, dialogue and functional core) and to propose techniques for reasoning about (and also verifying) the models. Applying formal description techniques can be beneficial during the various phases of the development process from the early phases (requirements analysis and elicitation) to the later ones including evaluation (testing).

Fig. 3 details the task allocation and integration of competencies in the process. It shows how analysis, design and development artifacts coming from software engineering, HCI and safety management can be used in a synergistic way to:

- Test if required usability, reliability and safety integrity levels are reached.
- Identify problems if required levels of usability, reliability and safety integrity have not been reached.

The very high-fidelity prototyping phase produces very high-fidelity prototypes of the system, complete and unambiguous description of the system and of safety software barriers as well as precise description of expected user behaviors. It enables fine tuning of these prototypes, descriptions and models to ensure that system's behavior will be fully compatible with user tasks and will prevent human error to endanger the system and its environment. All of these materials are the inputs of the next phase.

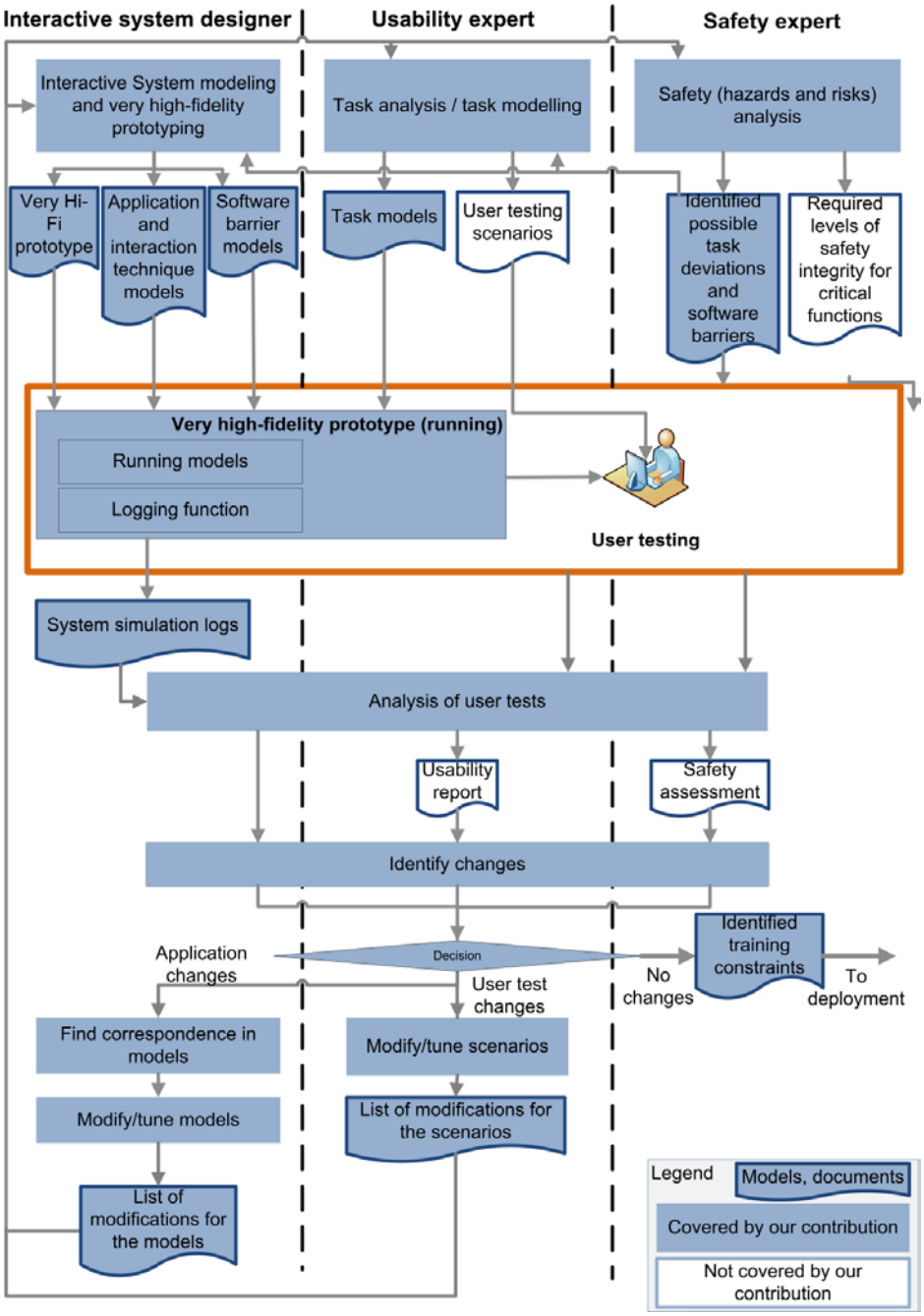


Fig. 3. Task allocation and integration of competencies in the process

4.5 Selective formal modeling

The safety analysis activity at the beginning of the development process has enabled to determine the boundaries between different parts of the system and the required safety integrity level (SIL, presented in section 2.1.3) for each of these parts. Associating a SIL to parts of the system allow to characterize the impact of a fault, failure or error on the system and its environment. When the required SIL is high, the use of models is easily justified and the use of formal models is recommended to verify and validate the system's behaviour avoiding fault occurrence by additional work at design time. For low SIL, standard development methods (involving either semi-formal approach like UML or programming languages) are adequate.

4.6 Training program development

The integration of the training program development as a phase of the system development process itself (disc III in Fig. 1) aims at producing the most optimized possible training program w.r.t. the designed system. Training program is based on the complete and unambiguous description of the system's behavior (very high-fidelity prototype and models). Model-based approaches provide a unique opportunity for integrating, in a unified iterative process, the four main artifacts i.e. tasks models, operational procedures, training scenarios and interactive system models required to be designed for usable, learnable and reliable command and control systems. This phase of the proposed development process has been detailed in [26].

4.7 Human error and safety management

The process is compliant with existing human error and safety management activities. Indeed, it supports human error and task deviation analysis (Task Analysis for Error Identification technique) [35] and software safety barriers modeling [20]. And, the task error models could be reused as safety management artifact (task error pattern) from one version of the system to the next and from one system to another. These task error patterns would contribute to the next hazards and risks analysis.

4.8 Traceability

The proposed process supports the traceability of requirements and design choices throughout the whole design process. Previous work on design rationale and traceability of requirements [25] can be integrated to this development process in order to support the record and analysis of design choices. Functional and nonfunctional requirements can be traced w.r.t. this design choices to perform coverage analysis. This work can be fully integrated within this approach and furthermore, as well as design artifacts (such as task models, scenarios, system models) can be bound to design choices, task models with error patterns and software safety models can also be bound to requirements and design choices.

4.9 Tools supporting the development process

As we mentioned in sections 2 and 3, supporting notation and tools for user tasks and system behavior are needed to handle usability within large-scale systems. In order to

apply the development process we used two existing notations, one for task modeling and one for system behavior modeling.

An expressive and scalable task modeling notation is required to support the design of a usable interactive application. HAMSTERS [24] (Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems) is a task modeling notation designed for representing the decomposition of human goals into activities (perceptive, cognitive, motor, interactive…). It supports the description of large-scale and complex tasks and its associated software tool (also called HAMSTERS) enables to edit task models and simulate their execution.

A formal notation is required to support the design of a dependable system, especially when high safety integrity levels have to be reached. ICO formalism [32] and associated Petshop IDE also provides support for specifying entirely an interactive system behavior (scalability) and for integrating software components to ICO formal models and then enabling:

- The integration of low safety integrity level components with high safety integrity level components.
- The integration of models of software safety barriers [20].

The last element highlighted by the development process is the need for performance evaluation for assessing the actual performance of the application (and the interaction techniques, disc 10 on Fig. 2). Here again the Petri net based description technique of ICOs is very useful as Petri nets are one of the very few formalism providing both verification and performance evaluation supports as demonstrated by [34].

Furthermore, the two tools, HAMSTERS and Petshop have been integrated in a software development framework, which provides support for correspondence matching between the two types of models and co-execution of the very high-fidelity prototype with the underlying system and task models [3].

5. Application of the development process to satellite monitoring and control applications

In order to ensure its feasibility, the proposed development process has been applied to large scale ground segment applications. This section summarizes the results from the application of this process and highlights how requirements for designed product and for design and development process have been met.

5.1 Context in which the process has been applied

This case study is part of the work that has been done for the TORTUGA[1] research and technology initiative, which aims at improving the reliability of both ground segment systems and users involved in the operation of such systems. Many models and prototypes have been produced for different industrial case studies accomplished

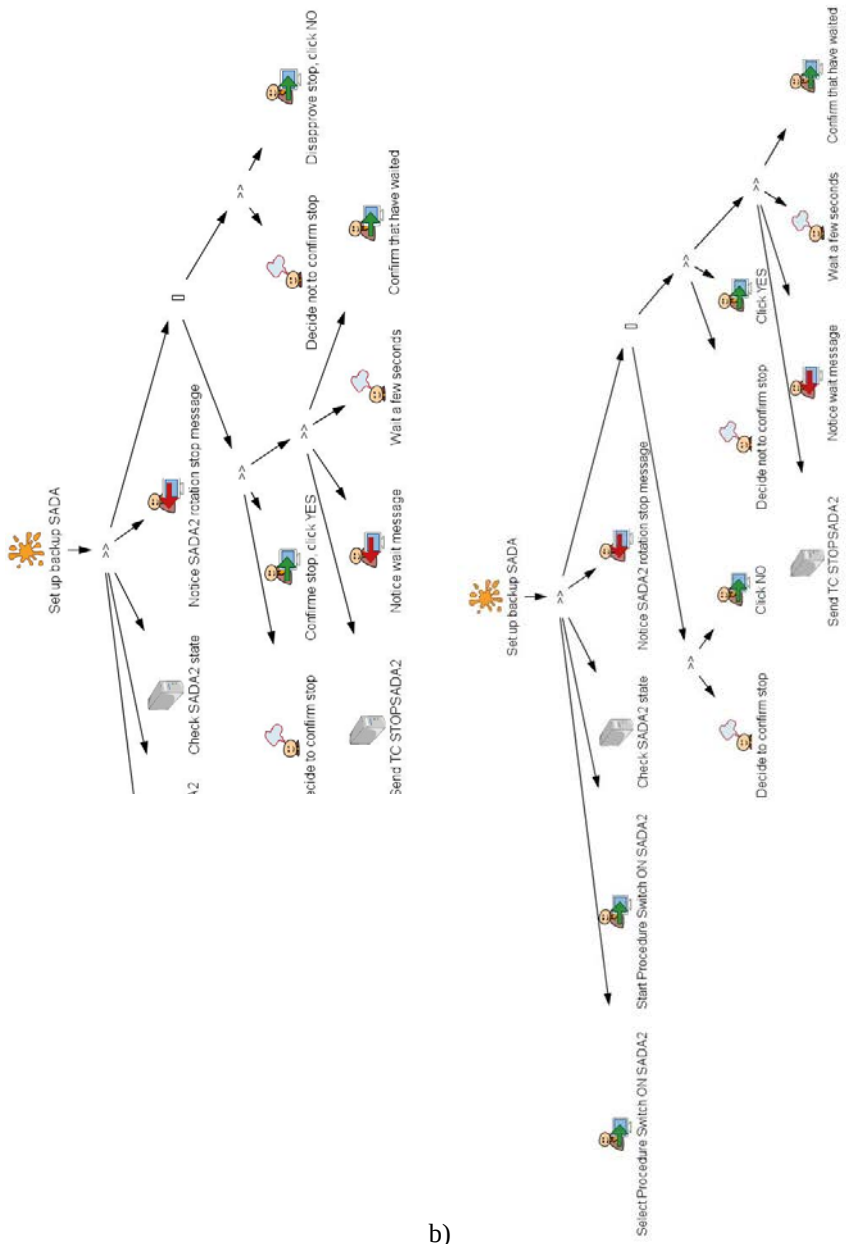
[1] http://www.irit.fr/recherches/ICS/projects/tortuga/

within this research project but only a few are shown, due to space constraints. Different entities are involved in the satellite application domain: the *space segment* (including the spacecraft) and the *ground segment* (made up of antennas for communication and the mission control system). Our focus is the operation control system. This system is in charge of maintaining the spacecraft in operation. During early validation phases of various ground segment applications, operators have encountered usability issues. CNES (French National Space Studies Center) Ground segment products department and Operations department agreed to study the feasibility of approaches developed within Tortuga project on a monitoring and control application of a ground segment. For this purpose, we applied the development process exposed in previous section to design and develop a very high-fidelity prototype for a new ground segment application and associated training sessions. We only provide here the excerpt necessary to discuss the application of the proposed development process. Indeed, tasks and operations of a mission control system are more numerous than what is presented here.

5.2 Artifacts produced during the application of the process

The first steps in applying this process have been to analyze existing documentation about the procedures operators have to follow, about the system they were using and their associated user manuals. Several observations of operators in command and control rooms have been conducted (for two different satellite missions). Operators have been interviewed and have been filled in questionnaires. The analysis of these artifacts led to produce a list of user needs as well as scenarios and task models. A human error analysis also led to produce task models of operators' potential interaction errors. Fig. 4 a) presents an extract of the task model describing the setup of a redundant solar panel of the satellite (SADA2) if a failure on the in charge solar panel (SADA1) has occurred. It indicates that the operator has to select a particular procedure "Switch ON SADA2" while using the ground segment application. Then, the operator has to activate the procedure. This will send a particular *Telecommand* to the satellite, which will start to rotate the solar panel. Then the operator will have to confirm the stop of the solar panel rotation. In this example, we focus on one type of error but an example of a complete case study of task analysis for error identification can be found in [35]. Human errors can occur while accomplishing this procedure to setup the redundant solar panel. For example, Fig. 4 b) presents the task model of erroneous actions performed by the operator in that case. From the Human Error Reference Table [35], an interference error [41] or associative-activation error [39] can occur if the operators click on "YES" while they had decided not to confirm stop or if they click on "NO" while they had decided to confirm stop (confirmation pop up in Fig.6).

Five low-fidelity prototypes have been produced and confronted to task models and to operators, then very-high fidelity prototyping phase started. This phase led to produce formal models (11 ICO models) of the ground segment application prototype and presentation part of the user interface (a screenshot of the UI is presented in Fig. 6), but also formal behavioral models (15 ICO models) of the procedures executed by



a) b)

the operators. Fig. 5 shows an excerpt of the ICO model corresponding to the procedure of switching to the redundant solar panel. This excerpt describes the system's behavior when it displays the message pop up to confirm that solar panel rotation will be stopped.

The presented notation and tool framework enables the binding of task models, system models and very Hi-Fi prototype. For example, task "Confirm stop, click

YES" in Fig. 4 a) corresponds to transition "msgYES_stopSADA2Rotation" in Fig. 5 and to "Yes" button of the confirmation pop up in Fig. 6.

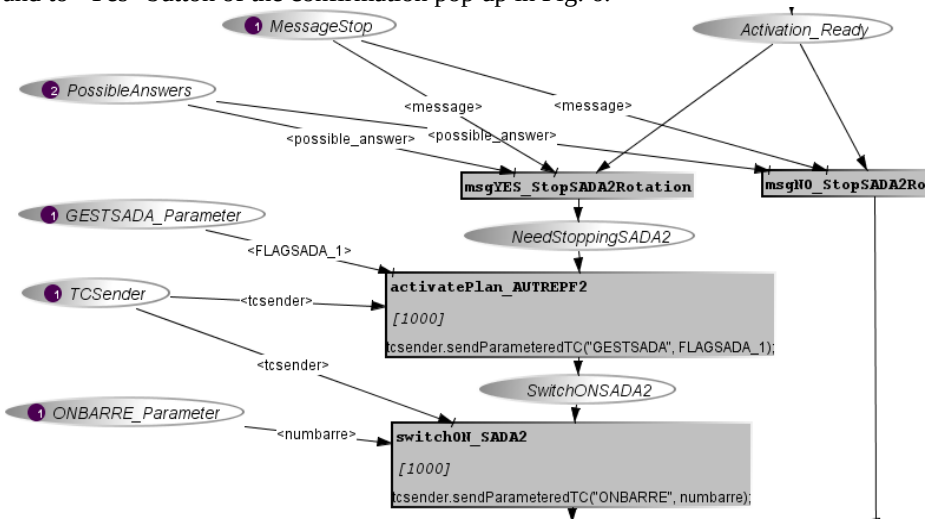


Fig. 5. Extract of the ICO model for switching to the redundant solar panel

Formal behavioral models of the system and of the procedures are put in correspondence with task models in the HAMSTERS-Petshop IDE. These models co-execute with the presentation part of the user interface (Fig. 6). Conformance and consistency between operators' tasks and prototype has been validated and training sessions have been developed and executed thanks to these produced artifacts.

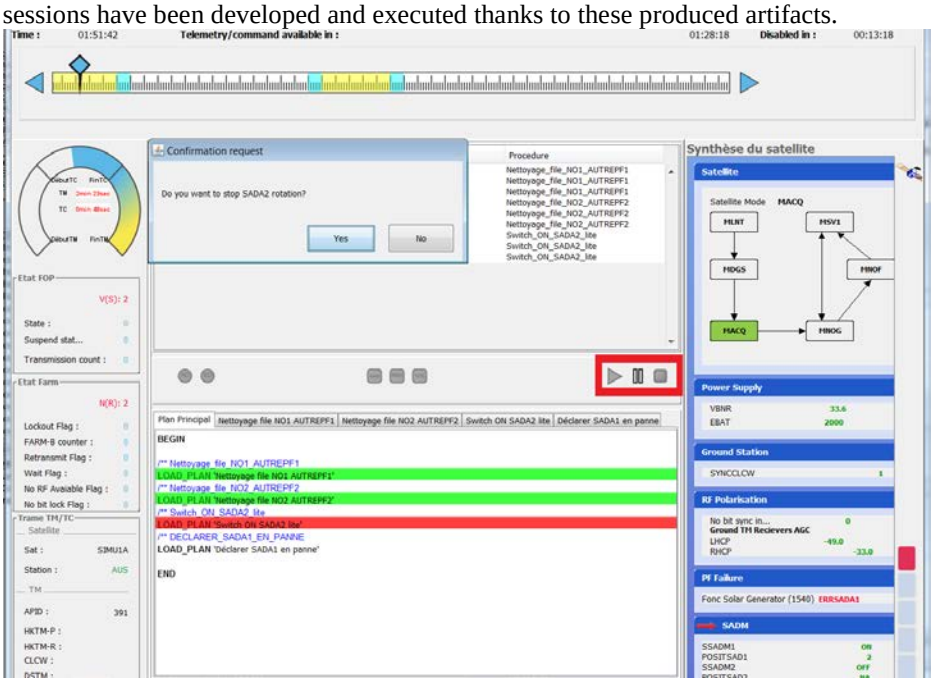


Fig. 6. Excerpt of the last iteration of the very high-fidelity prototype

Table 3 presents statistics about the artifacts produced while applying the development process. It is important to note that we only addressed in the project the most common functions in the ground segment applications deployed at CNES. We consider here neither mission-related functions nor specific aspect of the ground segments related to a specific satellite.

Table 3. Artifacts produced during the application of the development process

Artifacts of the ground segment application		Numerical values
Number of task models		20
Number of tasks per task model		12 to 20
Number of Low-Fi prototypes		5
Number of Hi-Fi prototypes		2
Number of ICO behavioral models		26
ICO models	places per model	15 to51
	transitions per model	10 to 36
	arcs per model	35 to 233
Number of training sessions		2
Number of scenarios per training session		3 to 5

5.3. How requirements are met throughout the development process

5.3.1 Presented in this article

The application of the development process to industrial projects proves the **scalability** of the proposed tool-supported approach. Task analysis and modeling supports the **usability and operability** of the designed and developed applications. Formal descriptions of the application behavior and of the operational procedures provide support for **reliability**. **Safety** concerns are addressed through the identification of human errors and tasks deviations (represented in the tasks models).

5.3.2. Not presented in this article

Going further in supporting **usability**, fine logging techniques, introduced in [34], enables quantitative analysis of user performance while using the system. These logging techniques can support in a very efficient way standard usability evaluation techniques especially when new and complex interaction techniques (such as multimodal ones) are considered [7].

Training is integrated as first class citizen in the development process. A detailed case study describing the training program development process and underlying concepts (such as Systematic Approach to Training), as well as the way our model-based approach can be used to leverage this process is described in [26]. Tasks models and system models are extremely useful for assessing the coverage of the training program and its adequateness to both the operators' tasks and system behavior.

Safety management phases are not described in details in this case study but complete examples can be found in previous work [35] [4] and can be fully integrated

in the proposed process. In particular, hazards and risk analysis techniques as well as safety integrity levels identification techniques [46] can be used within the proposed process. Additionally, It has been demonstrated in previous work that safety modeling [4] and software barrier modeling [5] phases can be plugged-in directly in the proposed process, as part of an integrated framework. **Training** is also linked to those aspects as many operations are defined for managing incidents and failures that by definition have a low probability of occurrence and thus require regular re-training in order not to be forgotten.

Traceability of the needs and requirements throughout the whole development process and design choices can be achieved [25] using adequate notations and tools which are complementary to those presented in this article.

Verification of the application can be achieved through applying formal techniques related to Petri nets (which are the underlying formalism of ICO) [36].

7. Conclusion

This paper proposed a development process for the design, implementation and evaluation of safety critical interactive systems. It deals explicitly with requirements that target interactive critical systems (requirements for the system and requirements for the development process). Beyond that it integrates the training program within the process providing a unique opportunity to deliver timely and with a perfect match both a system and its training material. It also explicitly describes the articulation between high-level SILs and low-level ones and thus provides integration for formal and informal approaches.

The excerpts from the industrial application of the process are presented to show what the various products of the development process are and do not aim at being comprehensive. While it is very difficult (not to say impossible) to demonstrate the validity of a development process we have tried to report at each step its advantages and its limitations. To go beyond this case study we would like to emphasize the fact that this contribution is built upon several contributions and comes from the outcomes of the implementation of these contributions in several critical application domains such as space ground segments applications, interactive cockpits of large civil aircrafts and air traffic control management.

Future work is directed towards multi-users interactions and activities as well as computer mediated communication with remote users thus handling explicitly the issues related to team work both in a local setting and remote. We also focus our work on the issues related to automation and more precisely at providing means for designing automation in a user-centered way and to ensure that safety critical requirement are taken into account throughout the development process.

Acknowledgments

This work has been partly funded by R&T CNES (National Space Studies Center) Tortuga R-S08/BS-0003-029 and Airbus under the contract CIFRE PBO D08028747-788/2008**.**

References

1. Aguinis H., Kraiger K. Benefits of Training and Development for Individuals and Teams, Organizations, and Society. Annual Review of Psychology, pp. 451-475, vol. 60, 2009.
2. Alonso-Rios D., Vasquez-Garcia A., Mosqueira-Rey E., Morey-Bonillo V. Usability: A Critical Analysis and Taxonomy. Intl. Journal of Human–Computer Interaction, 26(1), 53–74, 2010.
3. 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.
4. Basnyat S., Chozos N., Johnson C., Palanque P. Incident and Accident Investigation Techniques to Inform Model-Based Design of Safety-Critical Interactive Systems. DSV-IS 2005,pp. 51-66.
5. Basnyat S., Palanque P., Schupp B., Wright P. Formal socio-technical barrier modelling for safety-critical interactive systems design, Safety Science, vol. 45, issue 5, June 2007, pp. 545-565.
6. Bastide R., Navarre D. & Palanque P. A Tool-Supported Design Framework for Safety Critical Interactive Systems in Interacting with computers, Elsevier, vol. 15/3, pp. 309-328, 2003.
7. Bernhaupt R., Navarre D., Palanque P., Winckler M. Model-Based Evaluation: A New Way to Support Usability Evaluation of Multimodal Interactive Applications. In "Maturing Usability: Quality in Software, Interaction and Quality" Springer Verlag series on HCI, April 2007.
8. Bodart F., Hennebert A.-M, Leheureux J.-M. & Vanderdonckt J. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. in Human Factors in Computing Systems INTERCHI'93, Addison Wesley, 424-29, 1993.
9. Boehm B. A spiral model of software development and enhancement, ACM SIGSOFT Software Engineering Notes, v.11 n.4, p.14-24, August 1986.
10. Boehm B. A View of 20th and 21st Century Software Engineering. Invited talk, IEEE Int. Conf. on Software Engineering 2006, http://www.isr.uci.edu/icse-06/program/keynotes/boehm.html.
11. Carroll, J.M., Kellogg, W.A., and Rosson, M.B. The Task-Artifact Cycle. In J.M. Carroll, ed., Designing Interaction: Psychology at the Human-Computer Interface. Cambridge University Press, Cambridge, UK, 1991.
12. Collins, D. Designing Object-Oriented user interfaces. Readwoods City, CA: Benjamin/Cummings Publishing, Inc, 1995.
13. Curtis, B., Hefley, B. A WIMP no more: the maturing of user interface engineering. Interactions, 1(1), 1994.
14. Esteban O., Chatty S., Palanque P. Whizz'Ed: a Visual Environment for building Highly Interactive Software INTERACT,1995, Lillehammer, Norway, p. 121-127.

15. Eurocontrol, ESARR 6, Eurocontrol Safety Regulatory Requirement 6, Software in ATM Functionnal Systems, version 2.0, 2010.
16. European Organisation for Civil Aviation Equipment. (1992). DO-178B, Software Consideration in Airborne Systems and Equipment Certification. EUROCAE.
17. Göransson B., Gulliksen J., Boivie I. The Usability Design Process - Integrating User-Centered Systems Design in the Software Development Process. Software Process: Improvement and Practice, 8(2), 111-131, 2003.
18. Hartson, H., Hix, D. Human-computer interface development: concepts and systems for its management. ACM Computing Surveys, 21(1), 1989.
19. Hussain Z., Slany W., Holzinger A. Investigating Agile User-Centered Design in Practice: A Grounded Theory Perspective. USAB 2009, pp. 279-289, Springer LNCS.
20. International Standard Organisation. Space systems safety requirements. Part 1: System safety. ISO 16420-1, April 2004.
21. Jacob R. A Software Model and Specification Language for Non-WIMP User Interfaces. ACM Transactions on Computer-Human Interaction 6, n°. 1, 1-46, 1999.
22. Johnson, C. On the over emphasis of human error as a cause of aviation accidents: systemic failures and human error in US NTSB and Canadian TSB aviation reports 1996-2003, Ergonomics, 2006.
23. Martinie C., Palanque P., Navarre D., Winckler M. A formal approach supporting effective and efficient training program for improving operators' reliability. Safety and Reliability for managing Risk (ESREL 2010), p. 234-243.
24. Martinie, C., Palanque, P., Winckler, M. Structuring and Composition Mechanism to Address Scalability Issues in Task Models. Proceedings of the IFIP TC 13 INTERACT, LNCS Springer Verlag, 2011.
25. Martinie C., Palanque P., Winckler M., Conversy S. DREAMER: a design rationale environment for argumentation, modeling and engineering requirements. SIGDOC 2010: pp. 73-80.
26. Martinie C., Palanque P., Winckler M., Navarre D. & Poupart E. Model-Based Training: An Approach Supporting Operability of Critical Interactive Systems: Application to Satellite Ground Segments. In Proc. of EICS 2011, p. 53-62.
27. Mayhew DJ., The Usability Engineering Lifecycle, A practitioner's handbook for User Interface Design. Morgan Kaufmann Publishers, San Francisco, CA.
28. McDermid, J., & Ripken, K. Life cycle support in the Ada environment. ACM SIGAda Ada Letters, III (1), 1983.
29. Memon A. M. & Soffa M. L. Regression testing of GUIs. 9th European Software Engineering conf., pp. 118 – 127, 2003.
30. Memon A. M., Soffa M. L. Pollack M.E. Coverage criteria for GUI testing. 8th European Software Engineering conference, pp. 256 – 267, 2001.
31. Navarre D., Palanque P., Martinie C., Winckler M., Steere S. Formal Description Techniques for Human-Machine Interfaces - ModelS-Based Approaches for the Design and Evaluation of Dependable Usable Interactive Systems. Handbook of HMI, A Human-Centered Approach, USA, Ashgate.
32. Navarre, D., Palanque, P., Ladry, J., and 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 (Nov. 2009), pp. 1-56.
33. Norman, D. & Draper S. (eds.) (1986): User Centered System Design: New Perspectives on Human-Computer Interaction. Hillsdale, NJ, Lawrence Erlbaum Associates.

34. Palanque P., Barboni E., Martinie C., Navarre D., Winckler M. A model-based approach for supporting engineering usability evaluation of interaction techniques, EICS 2011, pp. 21-30, ACM SIGCHI.
35. Palanque, P., & Basnyat, S. Task Patterns for Taking Into Account in an Efficient and Systematic Way Both Standard and Erroneous User Behaviours. HESSD 2004, pp. 109-130,. Toulouse, France.
36. Palanque P., Bastide R. Verification of an Interactive Software by analysis of its formal specification INTERACT 1995, Lillehammer, Norway, pp. 191-197.
37. Palanque P., Bernhaupt R., Navarre D., Ould M., Winckler M. Supporting Usability Evaluation of Multimodal Man-Machine Interfaces for Space Ground Segment Applications Using Petri net Based Formal Specification. Ninth International Conference on Space Operations, Rome, Italy, June 18-22, 2006. CD-ROM proceedings.
38. Paternò F. Santoro C. Spano L. D. MARIA: a universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. In: ACM Transactions on Computer-Human Interaction, vol. 16 (4) article n. 19. ACM, 2009.
39. Preece J., Rogers Y., Sharp H., Benyon D., Holland S., Carey T. Human-Computer Interaction. Addison-Wesley, UK.
40. Rauterberg M. An Iterative-Cyclic Software Process Model. International Conference on Software Engineering and Knowledge Engineering. Capri, Italie: IEEE, 1992.
41. Reason J. Human Error, Cambridge University Press.
42. Rettig M. 1994. Prototyping for tiny fingers. Commun. ACM 37, 4 (April 1994), 21-27.
43. Royce W. Managing the Development of Large Software Systems. IEEE Wescon, pp 1-9, 1970.
44. Salas E., Cannon-Bower J. The Science of Training: A Decade of Progress. Ann. Review of Psychology, pp. 471-499, 2001.
45. Seffah A., Donyaee M., Kline R.B., Padda H.K. Usability measurement and metrics: A consolidated model. Journal of Software Quality Control, vol. 14, issue 2, June 2006.
46. Storey, N. Safety-critical computer systems. Addison-Wesley, 1996.
47. Schwaber K. Agile Project Management with Scrum, Microsoft Press, February 2004.
48. Sy D., Miller L. Optimizing Agile User-centred design. In *CHI '08 extended abstracts on Human factors in computing systems* (CHI EA '08). ACM, New York, NY, USA, 3897-3900..