Task Model-Based Systematic Analysis of Both System Failures and Human Errors

Célia Martinie, Philippe Palanque, Racim Fahssi, Jean-Paul Blanquart, Camille Fayollas, Christel Seguin

Task Model-Based Systematic Analysis of Both System Failures and Human Errors

C. Martinie, P. Palanque, R. Fahssi, J.-P. Blanquart, C. Fayollas, and C. Seguin

***Abstract*—The overall dependability of an interactive system is one of its weakest components, which is usually its user interface. The presented approach integrates techniques from the dependable computing field and elements of the user-centered design. Risk analysis and fault-tolerance techniques are used in combination with task analysis and modeling to describe and analyze the impact of system faults on human activities and the impact of human deviation or errors on system performance and overall mission performance. A technique for systematic analysis of human errors, effects, and criticality (HEECA) is proposed. It is inspired and adapted from the Failure Mode, Effects, and Criticality Analysis technique. The key points of the approach are: 1) the HEECA technique combining a systematic analysis of the effects of system faults and of human errors; and 2) a task modeling notation to describe and to assess the impact of system faults and human errors on operators' activities and system performance. These key points are illustrated on an example extracted from a case study of the space domain. It demonstrates the feasibility of this approach as well as its benefits in terms of identifying opportunities for redesigning the system, redesigning the operations, and for modifying operators' training.**

***Index Terms*—Human error (HE), risk analysis, system failure, task modeling.**

I. Introduction

THE overall dependability of an interactive system is the one of its weakest component, and there are many components in such systems ranging from the operator processing information and physically exploiting the hardware (input and output devices), interaction techniques, to the interactive application and possibly the underlying noninteractive system being controlled. This paper proposes an approach integrating these aspects in order to address system and human dependability altogether. These two aspects of dependability are usually dealt with separately as the research contributions come from different and usually unrelated scientific communities. In the dependable computing community, techniques have been proposed to cope with the impact of system failures and to assess it in a precise manner, but operators' behavior remains outside of the techniques. In the human reliability and in the human–computer interaction (HCI) communities, approaches have been proposed demonstrating the suitability of task modeling techniques to address system and human dependability analysis. This paper presents an integrated approach taking into account both system failures and human errors (HEs) while designing interactive systems. This approach aims at leveraging existing techniques in the fields of dependable computing, human reliability assessment, and HCI. The proposed technique also aims at providing complete and unambiguous task descriptions, which support fine-grain analysis of both human and system aspects.

This paper is structured as follows. Section II presents related work focusing on human-centered approaches to dependability. Section III provides a brief review of types of system failures and HEs but also exhibits a new type of source of errors namely interaction errors. Section IV presents a task model-based stepwise process to describe and analyze the impact of system faults and HEs in an integrated manner. A case study from the space domain follows that section, while last section concludes this paper.

II. Related Work

Existing approaches demonstrate the suitability of task modeling to address system and human dependability analysis, but system side and human side are usually addressed separately. Erroneous behavior described in task models can be used to evaluate the impact of an HE on the system as proposed in [5]–[7], [12]. Mutant task specifications have also been proposed to analyze the ability of the system to remain safe if a user performs deviated tasks on the system [33]. Task analysis has already been employed as an error identification technique [2] but with fewer task types (with respect to the notation used in this paper) and not for providing support for describing required information to perform the task. A model-based technique that uses insertion of HEs into task models in order to evaluate the usability of the system and to inform design has been proposed by Paterno and Santoro [27]. CREAM [14], HEART [32], and THERP [30] are human reliability assessment techniques that are based on task analysis. They provide support to assess the probability of occurrence of HEs. Their limitation is that the analysis does not go further than generic task types. They do not provide support to assess the possibility of a failure or error for human system interactions and as a consequence do not provide support to analyze in detail their potential impact. In the field of HCI, the THEA technique [28] helps to anticipate interaction failures, but it does not provide support for an integrated analysis of human and system dependability.

C. Martinie, P. Palanque, R. Fahssi, and C. Fayollas are with the Toulouse Institute of Computer Science Research, University Paul Sabatier Toulouse 3, 31062 Toulouse Cedex, France (e-mail: martinie@irit.fr; palanque@irit.fr; fahssi@irit.fr; fayollas@irit.fr).

J.-P. Blanquart is with Airbus Defence and Space, Toulouse, France (e-mail: jean-paul.blanquart@astrium.eads.net).

C. Seguin is with the ONERA lab, Toulouse, France (e-mail: Christel. Seguin@onera.fr).

We focus on human performance and on task recovery whether the deviated task is performed upon system failure and/or upon HE. We extend the work in [25] that proposes a task analysis for error identification technique. This technique can be used to identify potential HEs during routine activities as well as during failure detection and recovery activities. In [25], information, devices, and objects required to perform a task were not represented in the task models. Thus, it did not provide support to assess performance at the information level. This made impossible to reason about workload aspect of operator performance.

III. System Failures and Human Errors

System failures and HEs may both lead to problematic situations, while a system is in operation. However, as stated in Section I, they are generally addressed by different scientific communities and studied in an independent way, even though both contribute to the dependability level of the system under consideration. This section describes how the dependability community deals with system failures and how the human factor community deals with HEs.

A. Dealing With System Faults

In the fields of dependable computing and safety analysis, one can find fault taxonomies, methods for identifying system faults, methods to analyze their potential impacts, and techniques to remove them. Techniques for dealing with system faults have been proposed, and current state of the art in the field identifies four different ways to increase a system's reliability [1], [8].

1) Fault avoidance: preventing the occurrence of faults by construction (usage of design and development methods that avoid the integration of faults).
2) Fault removal: reducing the number of faults that can occur (by verification of properties).
3) Fault forecasting: estimating the number, future incidence, and likely consequences of faults (usually by statistical evaluation of the occurrence and consequences of faults).
4) Fault tolerance: avoiding service failure in the presence of faults via fault detection and fault recovery. Fault detection corresponds to the identification of the presence of faults, their type, and possibly their source. Fault recovery aims at transforming the system state that contains one or more faults into a state without fault so that the service can still be delivered.

Fault avoidance and fault removal can be attained by the formal specification of the interactive system behavior provided all the aspects of interactive systems are accounted for including device drivers' behaviors, graphical rendering and events handling and a Petri net based approach dealing with these aspects can be found here [22]. However, due to this software/hardware integration faults might occur at runtime regardless the effort deployed during design phases. To increase the system reliability concerning runtime faults, we have previously proposed [23] ways to address both fault tolerance and fault mitigation for safety critical interactive systems. Fault recovery is usually achieved by adding redundancy or diversity using multiple versions of the same software or by fault mitigation: reducing the severity of faults using barriers or healing behaviors [24].

TABLE I
Severity for Different Severity Categories and Failure Effect

Severity category	Severity numbers	Failure effect
catastrophic	4	possible death or system loss
critical	3	possible major injury or system damage
major	2	possible minor injury or mission effectiveness degradation
negligible	1	requires system maintenance, but does not pose a hazard to personnel or mission effectiveness

TABLE II
Probability Levels, Limits, and Numbers

Level	PN
Probable	4
Occasional	3
Remote	2
Extremely remote	1

In several application domains (such as aeronautics, aerospace, and automotive industry), these dependable computing techniques are applied using an approach, which is based on Failure Modes, Effects, and Critical Analysis (FMECA) [31]. FMECA is a risk identification technique that focuses on the system components. It is defined as "a procedure or technique to analyze each potential failure in a system to determine the results or effects thereof on the system and classify each potential failure mode depending to its severity." FMECA is also used to define preventive maintenance actions, operational constraints, and other relevant information and activities necessary to minimize the risk of failure [11]. There may be slight differences in applying the FMECA technique depending on the application domain and industrial context. Our work is based on the aerospace-domain standards [11]. In this context, the FMECA analysis process consists of the following steps.

1) Definition of product (hardware or function) to analyze. Complete definition and functional descriptions.
2) Prepare functional and reliability block diagrams, which illustrate the operation, interrelationships, and interdependences of the items which constitute the product.
3) Identify all potential failure modes for each item and investigate their effect on the item under analysis.
4) Evaluate each failure mode in terms of the worst potential consequences and assign a severity category (categories and corresponding severity number (SN) are described in Table I).
5) Assess the probability of occurrence of each identified failure mode and assign a criticality number (CN) using $CN = SN \times PN$ (where PN is the probability number, described in Table II). A probability of occurrence belongs to a probability level which matches a PN.
6) Identify failure detection methods and existing compensating provisions for each failure mode.
7) Identify for all critical items corrective design or other actions (such as operator actions) required to eliminate the failure or to mitigate or to control the risk.

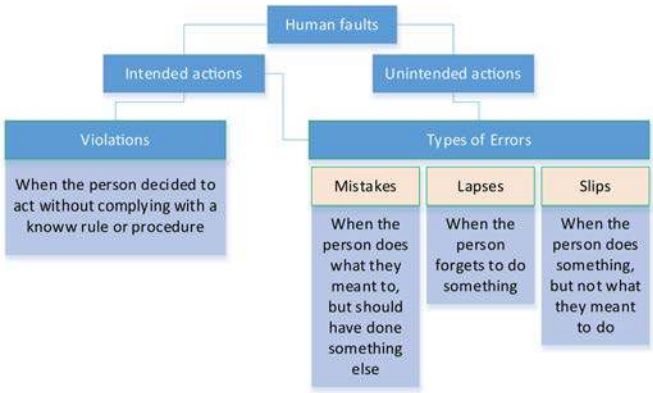

Fig. 1. HEs and faults (from [29]).

8) Document the analysis and summarize the results and the problems that cannot be solved by the corrective actions. Record all critical items into a dedicated table which shall be an input to the overall project critical item list.

Step 3 is related to fault forecasting. Steps 6 and 7 are related to fault removal, avoidance, and tolerance.

B. Dealing With Human Errors

In human factors ways of identifying, classifying and preventing errors from occurring have been extensively studied. Several taxonomies of HEs have been proposed such as [29] and [13]. Fig. 1 provides a summary of these types of errors.

The classification in Fig. 1 exhibits two main types of faults: 1) a violation that corresponds to the operator intentionally behaving in an unexpected way; and 2) an error when the unexpected outcome is not produced intentionally by the operator. When operating an interactive computing system, operator faults can be avoided by acting at three different levels:

1) *System level:* for instance by checking that a value provided by the operator is within an acceptable range. In that case, the system monitors the operator's activity and aims to prevent the occurrence of faults. Sometimes such protections lay outside of the system and are then usually called barriers [13].
2) *Interaction level:* for instance by preventing the entry of wrong values by only allowing users to select valid ones via radio buttons.
3) *Operator level:* for instance by providing thorough and adequate training, informing users about the impact and consequences of decisions.

More refined descriptions of HEs offer more refined ways of forecasting, preventing, tolerating or even avoiding them. For instance, a HE reference table is proposed in [4]. It refines skill-level errors from [29] and identifies means of preventing them from reoccurring either by redesigning the system or by adding barriers to it.

C. Integration Issues

Fig. 2 presents the taxonomy of faults that cover both system faults and human faults. It separates errors that may occur at development time or at operation time and is adapted from [1]. This taxonomy aims at being exhaustive covering even the notion of malicious behaviors that are illustrated in Fig. 2. However, this taxonomy classifies faults as if they were unrelated and occurring in an independent manner, and current methods, techniques, and tools address them independently and promote different treatment. This paper proposes to integrate system fault analysis and HE analysis. It also proposes to use the concept of criticality levels of HEs as it is used in the field of dependable computing.

IV. Integrated Approach to Account for System Failures and Human Errors

A. Stepwise Process to Account for System Failures and Human Error

The process for taking into account both system faults and HEs at design and development time is illustrated in Fig. 3. The proposed process is decomposed in seven phases:

1) task analysis and modeling (similar to steps 1 and 2 of the FMECA analysis process);
2) filtering out tasks and actions depending on the type of analysis to be performed;
3) effects and criticality analysis for HEs and system failure modes (similar to steps 3–5 of the FMECA analysis process);
4) inventory of the couples {activity node, criticality} and inventory of the additional tasks that would be needed to recover from system failures and/or HEs (which matches step 6 of the FMECA analysis process);
5) construction of enriched task models (models integrating potential system failures and HEs as well as articulatory tasks to recover from them);
6) construction of enriched task models (models integrating potential system failures and HEs as well as articulatory tasks to recover from them);
7) analysis of the impact of the system faults and HEs on the users' performance and on the global mission (system and organization);
8) identification of design alternatives and proposals for modifying users' tasks and/or system's functions (which matches steps 6 of the FMECA analysis process).

Phase 1 consists of analyzing and describing user's activities with the envisioned system similar to user-centered approaches for designing interactive systems. These activities are recorded in task models and several versions of the models can be produced to ensure that they are fulfilling the desired properties (and that they are compliant with the system's behavior and functions).

Phase 2 consists of filtering out the human tasks and actions as input to the Human Errors, Effects and Criticality Analysis (HEECA) and in filtering out system tasks as input for the FMECA. In this paper, the emphasis is on the proposed HEECA technique as the FMECA technique is already well described.

Phase 3, the HEECA technique, is described in next section. Phases 4–7 are described in the section dedicated to support for design and development.

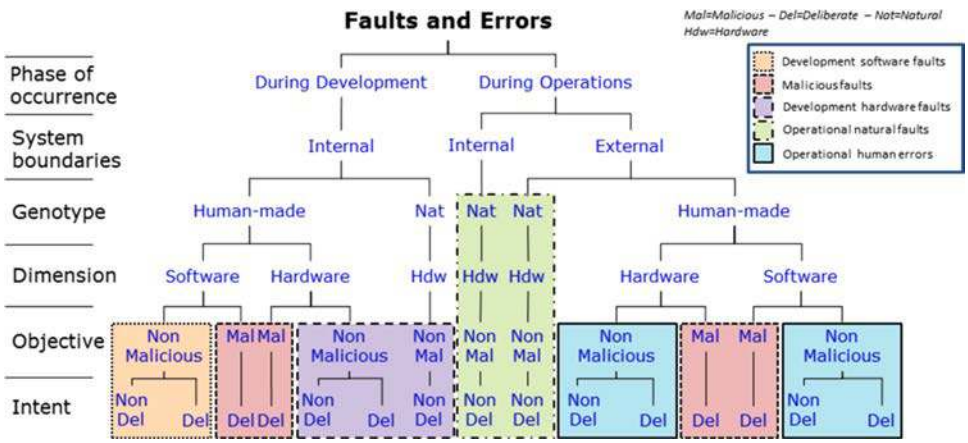

Fig. 2. Typology of faults in computing systems adapted from [1].

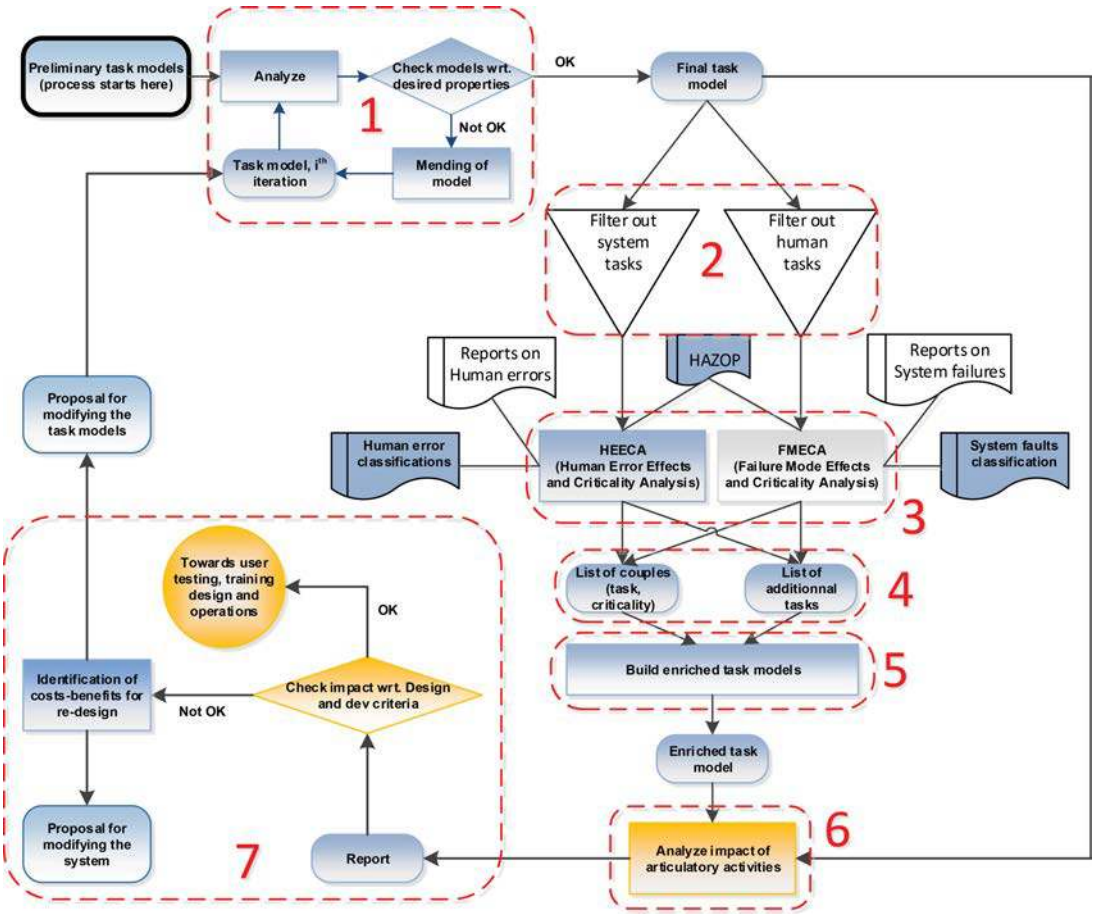

Fig. 3. Process to account for system failures and HE during the design and development of an interactive critical system.

B. Task Modeling With Human-Centered Assessment and Modeling to Support Task Engineering for Resilient Systems (Phase 1)

Task models support gathering and structuring data from the analysis of users' activities, and recording, refining, and analyzing information about users' activities. Several notations are available to describe tasks with varying expressiveness levels depending on targeted analysis, one of the most famous being CTT [21]. Human-Centered Assessment and Modeling to Support Task Engineering for Resilient Systems (HAMSTERS) is a tool-supported graphical task modeling notation for representing human activities in a hierarchical and ordered way. At the higher

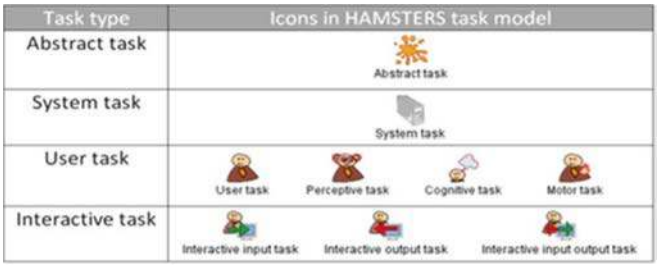

Fig. 4. High-level task types in HAMSTERS.

TABLE III
TEMPORAL ORDERING OPERATORS IN HAMSTERS

Operator type	Symbol	Description
Enable	T1 ≫ T2	T2 is executed after T1
Concurrent	T1 |||| T2	T1 and T2 are executed at the same time
Choice	T1 [] T2	T1 is executed OR T2 is executed
Disable	T1 [> T2	Execution of T2 interrupts the execution of T1
Suspend-resume	T1 |> T2	Execution of T2 interrupts the execution of T1, T1 execution is resumed after T2
Order Independent	T1 |=| T2	T1 is executed then T2 OR T2 is executed then T1

abstraction level, goals can be decomposed into subgoals, which can in turn be decomposed into activities. Output of this decomposition is a graphical tree of nodes. Nodes can be tasks or temporal operators.

Tasks can be of several types (see Fig. 4) and contain information such as a name, information details, and critical level. Only the high-level task type are presented here, but they are further refined (for instance the cognitive tasks can be refined in analysis and decision tasks) [7].

Temporal operators are used to represent temporal relationships between subgoals and between activities (see Table III). Tasks can also be tagged by temporal properties to indicate whether or not they are iterative, optional, or both.

HAMSTERS' notation and tool provide support for task-system integration at the tool level [3]. They provide support for:

1) automation design. The notation has been extended to help with the analysis of function allocation between human and system thanks to the refinement of cognitive tasks into analysis and decision subtypes of cognitive tasks [17] according to the Parasuraman et al. model of human information processing [26], [26];
2) structuring a large number and complex set of tasks introducing the mechanism of subroutines [20];
3) describing data that are required and manipulated [18] in order to accomplish tasks.

We propose to refine the following elements of notation: information, objects, and input/output devices. Fig. 5 recapitulates the existing notation elements as well as the refined ones. Information ("I:" followed by a text box) may be required for execution of a system task, but it also may be required by the user to accomplish a task.

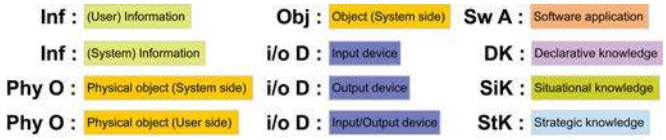

Fig. 5. Representation of objects, information and knowledge with HAMSTERS notation.

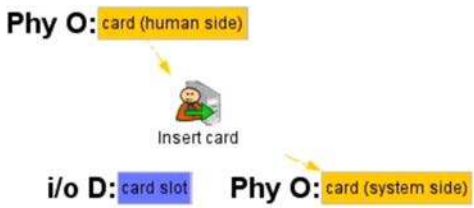

Fig. 6. Example of a task description with consumed and produced objects.

The notation element "Physical Object" ("Phy O:" followed by a text box) supports describing whether a user or a system task requires a particular physical object to be accomplished. The notation element "Object" ("O:" followed by a text box) is dedicated to the description of a software data object that is required by the system in order to accomplish a task. The notation element "Software Application" ("Sw A:" followed by a text box) provides support for the description of a software application that is required in order to accomplish a task. Furthermore, it is possible to describe how these notation elements are consumed and/or produced by a task. As illustrated in Fig. 6, an arrow incoming to a task means that the information or physical object (or object or software application or knowledge) is consumed, whereas an arrow outgoing from a task to an information or object means that it is produced by the task.

A precise description of users' activities (encompassing procedural and declarative aspects of these activities) is required to be able to systematically:

1) enumerate and record potential deviations and errors when the user is performing an action;
2) enumerate and record potential problems in the temporal ordering of actions;
3) enumerate and record potential problems in data and objects flow between operators and system (both in terms of inputs and outputs);
4) connect these potential deviations and errors to existing error taxonomies and thus identify already defined ways to handle them.

HAMSTERS notation is a key element of the proposed approach as it provides support to all of the above-mentioned points.

C. Support for Analysis: Human Errors, Effects, and Criticality Analysis (Phases 2 and 3)

We propose a technique inspired and adapted from FMECA to analyze in a systematic way the effects and criticality of HEs. The HEECA analysis process consists of several steps.

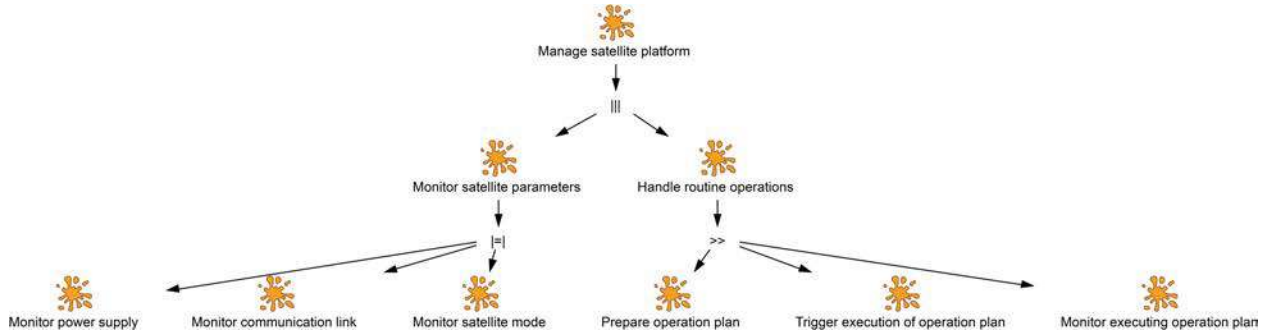

Fig. 7. HAMSTERS task model of PICARD satellite platform management (high-level tasks).

1) Definition of roles, task nodes, and action nodes to analyze. Complete descriptions have been performed during the task modeling phase. Their interrelationships are also described in the task models.
2) Identify all potential errors (using HE reference classifications) and deviations (using the HAZOP method [15]) for each item and investigate their effect on the item under analysis.
3) Prepare scenarios that illustrate potential errors and deviations and their consequences.
4) Evaluate each potential error or deviation in terms of the worst potential consequences (on the corresponding goal and on the mission) and assign a severity category (categories and corresponding SN are in Table I).
5) Assess the probability of occurrence of each identified potential error or deviation and assign a CN (which corresponds to the probability quantification steps of existing methods).

This process is integrated into the whole approach presented in this paper, i.e., precisely analyzing the impact of potential errors and deviations, as well as reporting them and proposing modifications corresponds to phases 4–7 (see Fig. 3).

D. Modeling Articulatory Activities (Phases 4 and 5)

Phase 4 aims at producing a list of couples (task, criticality) that will highlight which tasks have to be carefully examined to avoid critical issues. Depending on the criticality threshold that has been set for the mission, a subset of tasks extracted from the list of couples (task, criticality) can be selected.

These subsets of selected tasks integrate human activities that may be subject to errors or deviations (output of the HEECA), as well as system failures that may happen (output of the FMECA). For each entry of each subset, the sequence of additional actions required to recover is defined. These sequences of additional actions are then used to build enriched task models for each selected couple (task, criticality). Each of these models embeds a potential error or deviation of a task or system failure impacting a task. For each of these tasks, the task model is extended with the actions that have to be performed to recover from the error, the deviation, or the system failure.

E. Analyze Impact of Articulatory Activities (Phase 6)

This phase consists in quantifying the impact of errors and deviations or system failures on the user's goal, on the system and on the mission. Thanks to the produced task models, it is possible to estimate:

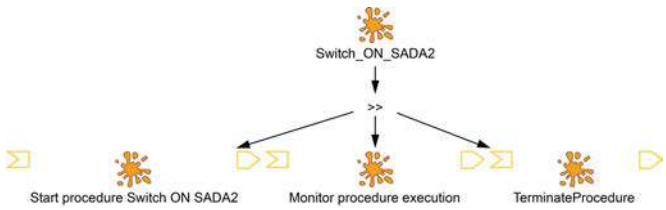

Fig. 8. Task model of the "Switch ON SADA2" task.

1) the number of actions for each type (cognitive, motor, perceptive actions, as well as interactive actions and system actions) that will have to be performed to recover from the error or deviation;
2) the number and nature of information (procedural, declarative) that will have to be handled compared to the case where the error or deviation may not happen;
3) the potentially nonreversible aspect of the error or deviation (when recovery is impossible).

F. Identify Cost Benefits for Redesign and Propose Modifications for User Tasks and/or System's Functions (Phase 7)

From the assessment of impact of errors and deviations and system failures on the users' tasks, system, and mission, it is then possible to identify ways of redesigning them and the feasibility of redesigning them with respect to resources constraints (such as users, time, and finances).

Dependable computing techniques (as the ones presented above) may be applied to the system, and we also propose to adapt those techniques for "removing" HEs.

In addition to and inspired from actions (dependable computing techniques) that can be taken after a FMECA, several actions may be taken in order to "remove" HEs.

1) HE avoidance: preventing the occurrence of HEs by construction (usage of design and development methods that avoid the possibility of a human making an error).
2) HE removal: reducing the number of errors that can happen (by verification of properties).
3) HE forecasting: estimating the number, future incidence and likely consequences of errors (usually by statisti-

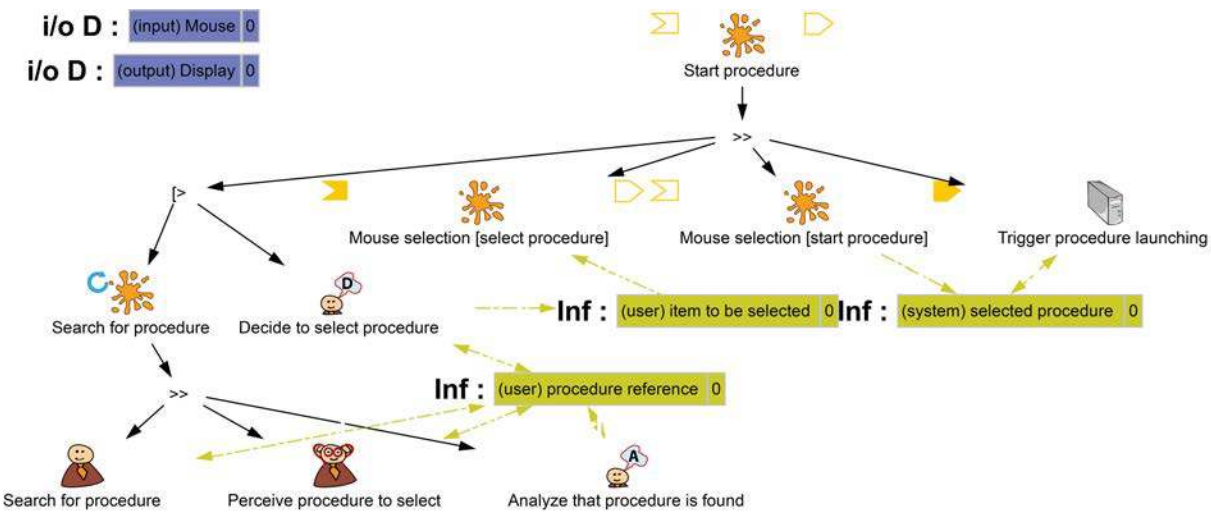

Fig. 9. Task model of "Start selection" task.

cal evaluation of the occurrence and consequences of errors).

4) HE tolerance: avoiding task failure in the presence of HEs via HE detection and/or adding barriers and/or mechanisms for HE recovery.

V. PICARD Satellite Case Study

The PICARD satellite dedicated to solar observation was launched by CNES in June 2010. We use a subset of it for our case study.

A. PICARD Satellite Ground Segment

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 (definition of operational procedures may be found in the ECSS-E-70-32A [10] standard).

Among the various ground segment applications used to manage the satellite platform, we focus on the ones that are used by controllers to ensure that the platform is functional. The platform has to be functional so that the mission (for which the satellite has been designed and developed) can be completed. The controllers of the PICARD satellite take turn in the command and control room (there is only one on duty at a time). Controllers have several applications (with their corresponding displays) for the monitoring activities and dedicated applications to manage the procedures and the telecommands plans.

B. Controller's Tasks Analysis and Modeling

Controllers are in charge of two main activities: observing periodically (i.e., monitoring) the vital parameters of the satellite and performing maintenance operations when a failure occurs. Depending on the satellite between thousands and tens of thousands parameters have to be monitored. The more frequent and relevant monitoring activities include observing: satellite mode, telemetry (measures coming from the satellite), sun array drivers statuses, error parameters for the platform, error parameters for the mission, power voltage (energy for the satellite), ground station communication status, and on-board computer main parameters. Fig. 7 presents the HAMSTERS task model of the high-level tasks for the management of PICARD satellite platform. It describes the main abstract tasks that controllers have to execute.

The whole set of refined models which includes low-level routine activities (such as in Fig. 9 that describes at a fine grain level the actions the controller has to perform) is not presented in this paper. To illustrate the proposed approach, we present here the particular task of executing an operational procedure. The operational procedure we use deals with setting up the redundant sun array driver assembly (SADA), which is named SADA2.

Fig. 8 presents the main tasks that have to be led to execute this procedure. First, the controller has to start the procedure, then to monitor its execution, and in the end to terminate the procedure. Each task leaf in this model is a subroutine and has a corresponding refined task model.

The "Start procedure" subroutine is refined in Fig. 9. Fine grain modeling of users' actions with an interactive system is bound to the interactive system interface. The task models are highly dependent on the way the information is presented and reachable in the user interface. In this case study, the software application used by controllers is a procedure manager (see Fig. 11). The controller can select a procedure from the list (top left widget in Fig. 11), and then, she/he can start the procedure by pressing the "Start Procedure" button.

The procedure ("Search for procedure" iterative task). Once the controller has decided to select the procedure, the search task will be disabled (operator "[>") and the next task will be to:

1) Select the procedure ("Mouse selection [select procedure]" task, of subroutine type, in Fig. 9)
2) Start the procedure ("Mouse selection [start procedure] task, of subroutine type, in Fig. 9)

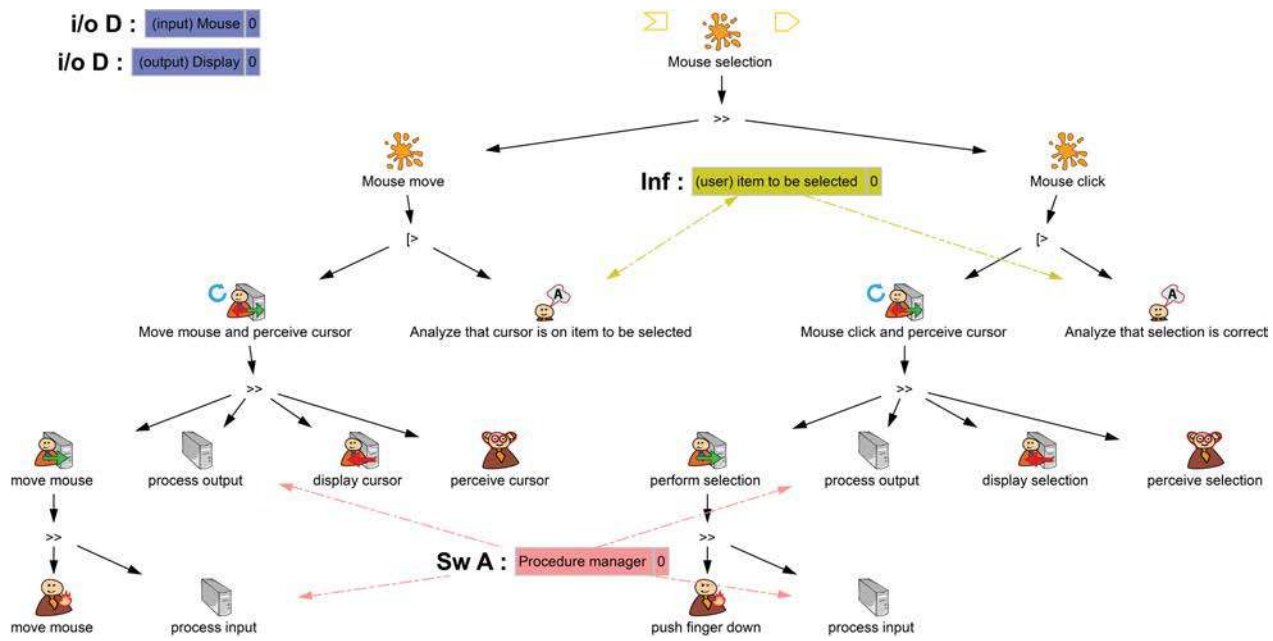

Fig. 10. Task model of "Mouse procedure" task.

3) At last, the system will start executing the procedure (system task in Fig. 9). This task model also describes which information is required to reach the goal of starting a procedure.
4) The information about procedure reference. This information is required to be able to search for it in the list and to analyze that the targeted procedure has been found in the list (Box "I: (user) procedure reference with incoming and outgoing arrows to "Search for procedure" user task, "Perceive procedure to select" perceptive task and "Analyze that procedure is found" cognitive analysis task).
5) The information about the item (of the list) to be selected. Once the controller has decided to select the procedure, she/he produces new information which is the information about the item to be selected (Box "I: (user) item to be selected" with incoming arrow from the "Decide to select procedure" cognitive decision task and with an outgoing arrow to the "Mouse selection [select procedure]" subroutine task).

Fig. 10 presents the "Mouse selection" task model. It describes the fine grain actions that have to be performed for selecting a graphical object with a mouse device and pointer. It also describes the required information to reach this goal.

C. Human Errors, Effects, and Criticality Analysis for the Task of Procedure Selection, Triggering, and Monitoring

Filtering out human actions from task models enables picking out the tasks and actions for which deviations and/or HEs may happen. The HEECA technique is then applied on these identified tasks and actions in order to systematically go through the potential issues and find out their criticality.

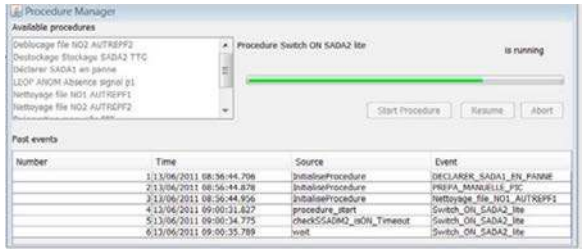

Fig. 11. Screenshot of the procedure manager software application.

Fig. 12 contains an extract from the HEECA table for the controller's task of driving the execution of a procedure. For the rest of the example, we focus on the potential error related in line 3 of this table (surrounded with a bold rectangle). In this line, a critical issue is pointed out and would be caused by a perceptual confusion error when selecting the procedure to be launched. This error is related to the declarative information about the item to be selected that the controller has in mind (as shown in Fig. 13). She/he may analyze that the good item in the list has been selected whereas it is not. As described by the scenario, procedures can have names that differ only by a few characters, which may cause perceptual confusion errors [29].

D. Identification of Couples {Task, Criticality} for the Task "Mouse Selection: [Select Procedure]"

In this example, the task "Mouse selection: select procedure" may have several criticality levels depending on the identified scenarios and may reach the highest criticality levels. An erroneous or deviated behavior during the mouse selection task may lead to delay in the mission and be tagged as critical.

For example, some procedures contain commands that put the satellite in a nonoperational mode. If this kind of proce-

#	Role	Scenario	Task node	Action node	Deviation (HAZOP)	Human error reference classifications	Error related to procedural or declarative information/ knowledge	Local effect	Effect on goal	Effect on mission	Severity	Proba	Criticality
1	Controller	The operator selects the procedure to trigger, but before the start it, he gets a call that made him start the procedure too late wrt. the satellite mode.	Start procedure SWITCH ON SADA2	Mouse selection: [start procedure]	Late	Not Applicable	Not applicable	Activity delayed	Task delayed	Delayed	Critical(3)	Remote (2)	6
2	Controller	The operator selects the procedure that is next to the good one, and he doesn't notice because he doesn't look the selected procedure on the display.	Mouse selection: [select procedure]	Perceive selection	No or not	Selectivity	Procedural	Another procedure is selected	Task interrupted	Delayed	Critical(3)	Remote (2)	6
3	Controller	The operator selects the procedure that is below the good one, and he doesn't notice because the both procedures have the same name by a few letters.	Mouse selection: [select procedure]	Analyze that selection is correct	Other than	Perceptual confusion	Declarative	Another procedure is selected	Task interrupted	Delayed	Critical(3)	Remote (2)	6
4	Controller	Operator select the wrong procedure but when he intends to change, he receives a call. After the call, the operator does not change the selection and starts the procedure.	Mouse selection: [select procedure]	Analyze that cursor is on item to be selected	Other than	Omissions following interruptions	Procedural	Another procedure is selected	Task interrupted	Delayed	Critical(3)	Remote (2)	6
5	Controller	The operator selects a procedure in a list, and before starting the procedure, he touches the "down" button without noticing, and the selected procedure is not the targeted one.	Start procedure SWITCH ON SADA2	Push finger down	Other than	Slip	Procedural	Another procedure is selected	Task interrupted	Delayed	Critical(3)	Remote (2)	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...
20	Controller	Operator select the wrong procedure but when he intends to change, he receives a call. After the call, the operator does not change the selection and starts the procedure. The unintentionnally selected procedure causes the mission to fail.	Mouse selection: [select procedure]	Analyze that cursor is on item to be selected	Other than	Omissions following interruptions	Procedural	Another procedure is selected	Task interrupted	Mission fails.	Catastrophic (4)	Remote (2)	8
		The operator selects a procedure in a list, and before starting the procedure, he touches the "down" button without	Start procedure	Push finger				Another	Task	Mission	Catastrophic		

Fig. 12. Extract from the HEECA table.

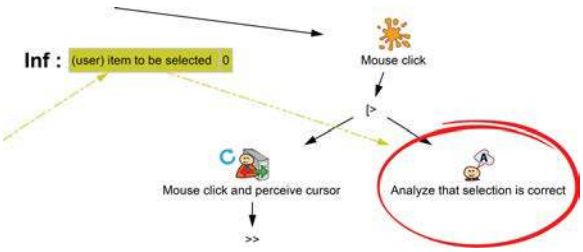

Fig. 13. Focus on the action node where an error may occur.

dure is selected unintentionally, the mission will not be able to be carried out anymore (at least for a certain period of time). The impact of this selection error can then be tagged as catastrophic.

E. Inventory of the Additional Activities in Case of the Perceptual Confusion Error Made While Analyzing the Selection

New versions of task models are built to describe the additional activities to be performed when an error or a deviation occurs. New versions can be built for each type of error. New version can also be built from FMECA tables to list all additional activities that would have to be performed to recover from a system failure [16].

In this example, we present the additional activities to recover from a perceptual confusion error during the "Mouse selection" task. Fig. 14 describes the additional activities in such a case. The controller will watch at the procedure execution ("Watch procedure execution" human task) and then analyze that the wrong procedure has been started ("Analyze wrong procedure has been started" cognitive analysis task). She/he may then decide to stop the procedure ("Decide to stop procedure" cognitive decision task) and stop it ("Stop procedure" interactive task). The next tasks (that are abstract and not described at a fine grain level in Fig. 14) are then to monitor the state of the SADAs and to determine if it is possible to start over the procedure and when. These two last abstract tasks have also been refined in user actions. They detail the heavy process of gathering expert's advice on the current satellite state and of preparing a plan (that may embed the execution of additional procedures) to determine how to recover the desired state.

F. Assessment of the Impact of Failures or Human Errors on Human Performance and Identification of Options to Redesign Ground Segment Applications and/or Controller's Tasks

From the task models enriched with additional activities that may have to be performed to recover from the perceptual confusion error (line 3 in HEECA table in Fig. 12), it is possible

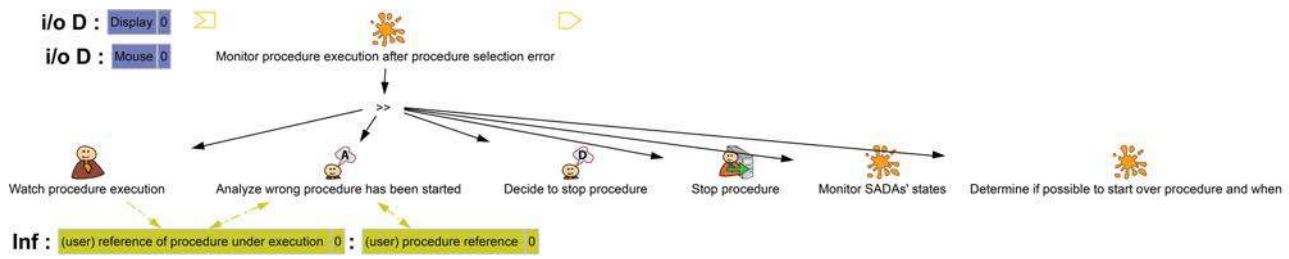

Fig. 14. Task model of "Monitor procedure execution" modified with additional activities required to recover from errors.

TABLE IV
NUMBER OF ACTIONS PER TYPE IN ORDER TO EXECUTE AND TERMINATE THE PROCEDURE TO SWITCH ON SADA2

	Cognitive analysis	Cognitive decision task	Motor task	Interactive input	Interactive output	Perceptive task
Without human error	7	3	4	4	9	9
With human error	16	5	10	12	25	25

to estimate the impact of this error on the controller's performance for the task of switching on the SADA2.

Table IV highlights that the perceptual confusion error done, while analyzing the item to be selected for the "Mouse selection: [select procedure]" task may have an impact on the controller's performance. Most of the actions she/he has to perform are at least doubled.

Following HE avoidance techniques proposed above, the following solutions could be proposed to avoid this type of error.

1) HE avoidance: Change procedure selection mean.
2) HE removal: Train the controller.
3) HE forecasting: Identify that operator is tired and /or carrying several threads of activities.
4) HE tolerance:
   a) Redundancy: Ask confirmation from several operators;
   b) Diversity: Use confirmation from different types/ trained users;
   c) Segregation: Have operations not co-located.
5) HE detection: Detect that the operator input is/was not appropriate;
6) HE mitigation: Trigger protection mechanisms in the system;
7) HE recovery: Provide procedures to recover from the error.

These design alternatives can then be chosen to be implemented in the system and/or to be used to modify the controllers' tasks. Another loop of the proposed process can then be performed to ensure that consequences of the potential HEs and system faults are under control.

VI. CONCLUSION

The presented approach integrates techniques from dependable computing and user-centered design in order to improve the reliability of interactive systems. Risk analysis and fault-tolerance techniques are used in combination with task analysis and modeling to describe and analyze the impact of system faults on human activities and the impact of human deviation or errors on system performance and more generally on mission performance. A technique for systematic analysis of HEs, effects, and criticality is proposed (HEECA). It is inspired and adapted from the FMECA technique.

By making explicit the operators' tasks, the information, and the objects that have to be handled while performing these tasks, this approach enables assessing the recovery cost from a system failure (i.e., to set the system in an acceptable state) but also from an HE. This recovery cost is expressed in terms of corrective actions that have to be performed in order set the system back to an error-free state.

This paper has presented the main phases of the process and described the HEECA method. However, a set of other benefits becomes reachable using such task models enhanced with HE descriptions, such as, for instance:

1) Some of the information explicitly represented in the task model might correspond to information that has to be stored in the operator's working memory (e.g., a flight-level clearance received by a pilot from an air traffic controller). The modeling approach would make explicit how much time (quantitative) but also how many actions have to be performed while keeping in mind such information.
2) The tasks and the related information might be located on specific devices. This is not the case for a space ground segment where the user interface used for monitoring is colocated with the one for triggering telecommands, but the possibility to represent that information in HAMSTERS enables assessing low-level complexity of tasks such as device localization or moving attention and activity between devices.

The outcome of the proposed process can also provide support for the traceability of requirements for training scenarios. It makes it possible to ensure that all critical tasks have been taken into account in the training program and that the operators have been trained for being able to recover from both user errors and system failures [19].

The presented analysis is performed informally and manually, but HAMSTERS models can be edited and simulated using the eponym tool. Performance analysis functionalities are currently being integrated exploiting contributions previously made for synergistic system-task execution [2] and training program assessment [9]. The proposed approach can be seen as a support for sharing information about risk analysis across

several domains. For example, it can be used as a bridge between safety experts that use HAZOP methods [15] and human factors experts using HE classifications and analysis methods.

REFERENCES

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.

[2] C. Baber and N. Stanton, "Task analysis for error identification," in *The Handbook of Task Analysis for Human–Computer Interaction*. Mahwah, NJ, USA: Lawrence Erlbaum Assoc., 2004, pp. 367–79.

[3] E. Barboni, J. F. Ladry, D. Navarre, P. Palanque, and M. Winckler, "Beyond modelling: An integrated environment supporting co-execution of tasks and systems models," in *Proc. ACM SIGCHI Eng. Interactive Comput. Syst.*, 2010, pp. 165–174.

[4] S. Basnyat, N. Chozos, and P. Palanque, "Multidisciplinary perspective on accident investigation," *Int. J. Rel. Eng. Syst. Safety*, vol. 91, pp. 1502–1520, 2006.

[5] R. Bastide and S. Basnyat, "Error patterns: Systematic investigation of deviations in task models," *Proc. 5th Int. Conf. Task Models Diagrams Users Interface Design*, 2006, pp. 109–122.

[6] M. L. Bolton, "Automatic validation and failure diagnosis of human-device interfaces using task analytic models and model checking," *Comput. Math. Org. Theory*, vol. 19, no. 3, pp. 288–312, 2013.

[7] M. L. Bolton and E. J. Bass, "Generating erroneous human behavior from strategic knowledge in task models and evaluating its impact on system safety with model checking," *IEEE Trans. Syst., Man, Cybern.*, vol. 43, no. 6, pp. 1314–1327, Nov. 2013.

[8] J. Bowen and V. Stavridou, "Formal methods, safety-critical systems and standards," *Softw. Eng. J.*, vol. 8, no. 4, pp. 189–209, 1993.

[9] European Cooperation for Space Standardization, "Space engineering, ground systems and operations," ECSS-E-70C & ECSS-E-70B Draft 4.2, 2008.

[10] European Cooperation for Space Standardization, "Space Engineering, Test and Operations Procedure Language," ECSS-E70-32A, 2006.

[11] European Cooperation for Space Standardization, "Space product assurance, failure modes, effects and criticality analysis (FMECA)," ECSS-Q-30-02A, 7 Sep. 2001.

[12] R. E. Fields, "Analysis of erroneous actions in the design of critical systems," D.Phil. dissertation, Dept. Comput. Sci., Univ. York, York, U.K., 2001.

[13] E. Hollnagel, *Barriers and Accident Prevention*. Surrey, U.K.: Ashgate, 2004.

[14] E. Hollnagel, *Cognitive Reliability and Error Analysis Method*. Oxford, U.K.: Elsevier Science, 1998.

[15] *Hazard and Operability Studies (HAZOP Studies)—Application Guide*, IEC 61882, 2001.

[16] C. Martinie and P. Palanque, "Fine grain modeling of task deviations for assessing qualitatively the impact of both system failures and human error on operator performance," in *Proc. AAAI Workshop Formal Verification Human Mach. Syst.*, 2014, pp. 27–32.

[17] C. Martinie, P. Palanque, M Ragosta, and E. Barboni, "Task-model based assessment of automation levels: Application to space ground segments," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2011, pp. 3267–3273.

[18] C. Martinie, P. Palanque, M. Ragosta, and R. Fahssi, "Extending procedural task models by explicit and systematic integration of objects, knowledge and information," in *Proc. Eur. Conf. Cognitive Ergonom.*, 2013, pp. 23–33.

[19] C. Martinie, P. Palanque, D. Navarre, M. Winckler, and E. Poupart, "model-based training: An approach supporting operability of critical interactive systems: Application to satellite ground segments," in *Proc. ACM SIGCHI Eng. Interactive Comput. Syst.*, 2011, pp. 53–62.

[20] C. Martinie, P. Palanque, and M. Winckler, "Structuring and composition mechanism to address scalability issues in task models," in *Proc. IFIP TC13 Conf. Human-Comput. Inter.*, 2011, pp. 589–609.

[21] G. Mori, F. Paternò, and C. Santoro, "CTTE: Support for developing and analyzing task models for interactive system design," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 797–813, Aug. 2002.

[22] D. Navarre, P. Palanque, J.-F. Ladry, and E. Barboni, "ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability," *ACM Trans. Computer-Human Inter.*, vol. 16, no. 4, p. 18, 2009.

[23] D. Navarre, P. Palanque, and S. Basnyat, "A formal approach for user interaction reconfiguration of safety critical interactive systems," in *Proc. Int. Conf Comp. Safety, Rel. Security*, 2008, pp. 373–386.

[24] S. Neema, T. Bapty, S. Shetty, and S. Nordstrom, "Autonomic fault mitigation in embedded systems," *Eng. Appl. Artif. Intell.*, vol. 17, no. 7, pp. 711–725, 2004.

[25] P. Palanque and S. Basnyat, "Task patterns for taking into account in an efficient and systematic way both standard and erroneous user behaviours," in *Proc. Int. Conf. Human Error, Safety Syst. Develop.*, 2004, pp. 109–130.

[26] R. Parasuraman, T. B. Sheridan and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Systems, Man, Cybern. A, Syst. Humans*, vol. 30, no. 3, pp. 286–297, May 2000.

[27] F. Paterno and C. Santoro, "Preventing user errors by systematic analysis of deviations from the system task model," *Int. J. Human Comput. Syst.*, vol. 56, no. 2, pp. 225–245, 2002.

[28] S. Pocock, M. Harrison, P. Wright, and P. Johnson, "THEA: A technique for human error assessment early in design," in *Proc. IFIP TC.13 Conf. Human-Comput. Inter.*, 2001, pp. 247–254.

[29] J. Reason, *Human Error*. Cambridge, U.K.: Cambridge Univ. Press, 1990.

[30] A. D. Swain and H. E. Guttman, *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications, Final Report, NUREG/CR-1278*, Washington, DC, USA: US Nuclear Regulatory Commiss., 1983.

[31] U.S. Department of Defense, "*Procedures for Performing a Failure Mode, Effects and Criticality Analysis*," MIL-STD-1629A, 1980.

[32] J. C. Williams "A data-based method for assessing and reducing human error to improve operational performance," in *Proc. IEEE 4th Conf. Human Factors Power Plants*, 1988, pp. 436–450.

[33] A. Yasmeen and E. L. Gunter, "Robustness for protection envelopes with respect to human task variation," in *Proc. IEEE Conf. Syst. Man Cybern.*, 2011, pp. 1809–1816.