Accounting for Organisational faults in Task Model Based Systematic Analysis of System Failures and Human Errors

Camille Fayollas, Célia Martinie, Philippe Palanque, Racim Fahssi
ICS-IRIT, University of Toulouse,
118 Route de Narbonne,
F-31062, Toulouse, France
name@irit.fr

Abstract. The overall dependability of an interactive system is the one of its weakest component which is usually its user interface. The presented approach integrates techniques from the dependable computing field and elements of user-centred design to provide a wider coverage of possible faults. Risk analysis and fault tolerance techniques are used in combination with task analysis and modelling to describe and analyse 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 is proposed (HEECA). It is inspired and adapted from the Failure Mode, Effects and Criticality Analysis (FMECA) technique. The key points of the approach are: a) the HEECA technique combining a systematic analysis of the effects of system faults and of human errors, b) a task modelling 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 re-designing the system, re-designing the operations and for modifying operators' training. Lastly, a discussion presents the main challenges for also taking into account organisational faults in an integrated way with the proposed approach.

1 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 non interactive 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' behaviour remains outside of the techniques. In the

human reliability and in the human computer interaction communities, approaches have been proposed demonstrating the suitability of task modelling techniques to address system and human dependability analysis. This paper presents an integrated approach taking into account both system failures and human errors while designing interactive systems. This approach aims at leveraging existing techniques in the fields of: dependable computing, human reliability assessment and human computer interaction. The proposed technique also aims at providing complete and unambiguous task descriptions which support fine-grain analysis of both human and system aspects. Finally, this paper proposes to extend this approach in order to take into account organisational faults.

The article is structured as follows. Section II provides a brief review of types of system faults and human errors. Section III presents a task model-based stepwise process to describe and analyse the impact of system faults and human errors in an integrated manner. A case study from the space domain is then presented in section IV providing concrete application of the process presented in section III. Section V is dedicated to the identification of ways of integrating organisational faults inside the approach while section VI concludes the paper.

2 System faults and human errors

This section presents a review of types of system faults and human errors it provides the underlying information that is used in the process described in section III.

2.1 Considering system faults

2.1.1 Development faults

When considering the faults that can impair interactive computing systems, the first kind of faults that come in mind are the software development faults. These faults are introduced by a human (the developer) during the system development. They can be, for instance, bad designs or programming errors and they result in software defects [5] that can lead to software failures. In the area of software engineering, methods and techniques have been introduced to prevent such faults and include (formal methods, structured programming, OO programing…). In the HCI community, a lot of work has been carried out for the prevention and removal of development software faults including software architectures (e.g. [2]), formal description techniques and verification (e.g. [17]), testing (e.g. [3]) or the use of debugging tools to remove human made faults.

2.1.2 Natural faults

In the domain of fault-tolerant systems, empirical studies have demonstrated (e.g. [18]) that software crashes may occur even though the development of the system has been extremely rigorous. One of the many sources of such crashes is called natural faults [1] triggered by alpha-particles from radioactive contaminants in the chips or neutron from cosmic radiation. A higher probability of occurrence of faults [27] concerns systems deployed in the high atmosphere (e.g. aircrafts) or in space (e.g. manned spacecraft [8]). Furthermore the evolution of modern IC components may lead in the next future to a higher probability of physical faults in operation. For instance, the recommendation for avionics systems is 100 FITs over 25 years lifetime, however, the current Deep Sub-Micron (DSP) technology may lead to a failure rate up to 1000 FITs, only during 5 years operational life time [25]. This is major worry in the avionics industry since this tendency has two bad sided effects, i) the reduction of the life time of the systems and ii) the increase of the failure rate due to hardware faults. Such natural faults demonstrate the need to go beyond classical fault avoidance at development time (usually brought by formal description techniques and properties verification) and to identify all the threats that can impair interactive systems.

2.2 Considering human errors

Several contributions in the human factors domain deal with studying internal human processes that may lead to actions that can be perceived as erroneous from an external view point. In the 1970s, Norman, Rasmussen and Reason have proposed theoretical frameworks to analyse human error. Norman, proposed a predictive model for errors [20], where the concept of "slip" is highlighted and causes of error are rooted in improper activation of patterns of action. Rasmussen proposes a model of human performance which distinguishes three levels: skills, rules and knowledge (SRK model) [21]. This model provides support for reasoning about possible human errors and has been used to classify error types. Reason [22] takes advantages of the contributions of Norman and Rasmussen, and distinguishes three main categories of errors:

1. Skill-based errors are related to the skill level of performance in SRK. These errors can be of one of the 2 following types: a) Slip, or routine error, which is defined as a mismatch between an intention and an action [20]; b) Lapse which is defined as a memory failure that prevents from executing an intended action.
2. Rule-based mistakes are related to the rule level of performance in SRK and are defined as the application of an inappropriate rule or procedure.

3. Knowledge-based errors are related to the knowledge level in SRK and are defined as an inappropriate usage of knowledge, or a lack of knowledge or corrupted knowledge preventing from correctly executing a task.

At the same time, Reason proposed a model of human performance called GEMS [22] (Generic Error Modelling System), which is also based on the SRK model and dedicated to the representation of human error mechanisms. GEMS is a conceptual framework that embeds a detailed description of the potential causes for each error types above. These causes are related to various models of human performance. For example, a perceptual confusion error in GEMS is related to the perceptual processor of the Human Processor model [4].

Causes of errors and their observation are different concepts that should be separated when analysing user errors. To do so, Hollnagel [9] proposed a terminology based on 2 main concepts: phenotype and genotype. The phenotype of an error is defined as the erroneous action that can be observed. The genotype of the error is defined as the characteristics of the operator that may contribute to the occurrence of an erroneous action.

These concepts and the classifications above provide support for reasoning about human errors and have been widely used to develop approaches to design and evaluate interactive systems [26]. As pointed out in [20] investigating the association between a phenotype and its potential genotypes is very difficult but is an important step in order to assess the error-proneness of an interactive system.

3 An integrated approach to account for system faults and human errors

This section presents the proposed approach to take into account for system failures and human errors at design and development time. It is composed of a stepwise process as well as modelling notation and tools [13].

3.1 A stepwise process to account for system faults and human error

The process for taking into account both system faults and human errors at design and development time is illustrated in Figure 3. The proposed process is decomposed in 7 phases:

1. Task analysis and modelling (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 human errors 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 human errors (which matches step 6 of the FMECA analysis process).
5. Construction of enriched task models (models integrating potential system failures and human errors as well as articulatory tasks to recover from them).
6. Analysis of the impact of the system faults and human errors on the users' performance and on the global mission (system and organization).
7. Identification of design alternatives and proposals for modifying users' tasks and/or system's functions (which matches steps 6 of the FMECA analysis process).

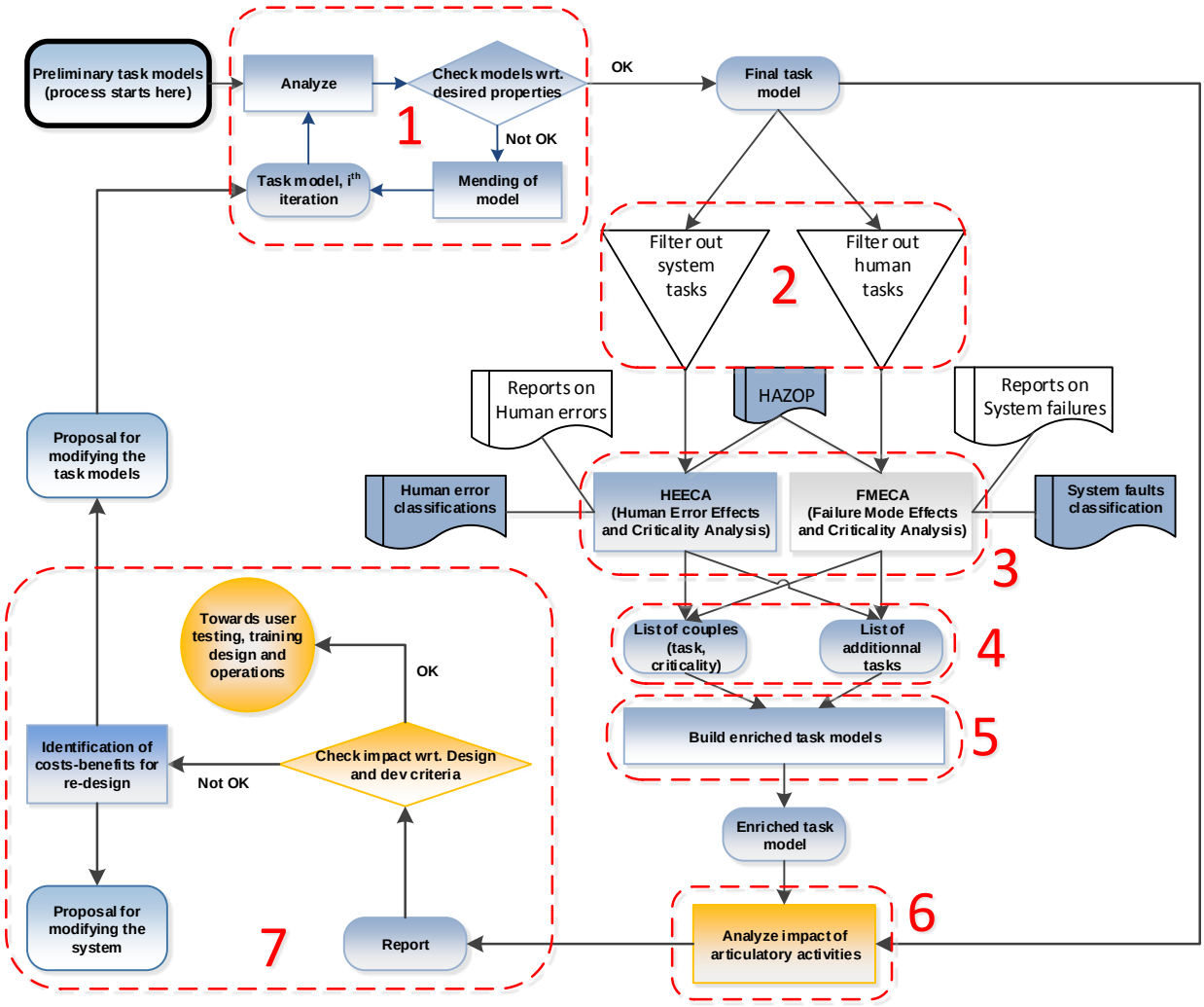

Figure 1. Process to account for system faults and human error during the design and development of an interactive critical system

3.2 Task modelling with HAMSTERS

HAMSTERS (Human – centered Assessment and Modeling to Support Task Engineering for Resilient Systems) is a tool-supported graphical task modelling notation for representing human activities in a hierarchical and structured way.

3.2.1 HAMSTERS notation

At the higher abstraction level, goals can be decomposed into sub-goals, which can in turn be decomposed into activities. Output of this decomposition is a graphical tree of nodes that can be tasks or temporal operators. Tasks can be of several types (cognitive, interactive, abstract…) and contain information such as a name, information details, and criticality level. Only the single user high-level task types are presented here but they can be further refined. For instance the cognitive tasks can be refined in Analysis and Decision tasks [12] and collaborative activities can be refined in several task types [10].

Temporal operators (based on LOTOS) are used to represent temporal relationships between sub-goals and between activities.

Tasks can also be tagged by properties to indicate whether or not they are iterative, optional or both. The HAMSTERS notation is supported by a CASE tool for edition and simulation of models. This tool supported notation also provides support for structuring a large number and complex set of tasks introducing the mechanism of subroutines [11], sub-models and components [7]. Such structuring mechanisms allow describing large and complex activities by means of task models. These structuring mechanisms enables the breakdown of a task model in several ones that can be reused in the same or different task models.

HAMSTERS expressive power goes beyond most other task modelling notations particularly by providing detailed means for describing data that is required and manipulated [15] in order to accomplish tasks.

3.2.2 Relationship between HAMSTERS notation elements and genotypes

All of the above notation elements are required to be able to systematically identify and represent human errors within task models. Indeed, some genotypes (i.e. causes of human errors) can only occur with a specific type of task or with a specific element in a task model described using HAMSTERS. This relationship between classification of genotypes in human error models and task modelling elements is not trivial. For this reason, Table 1 presents the correspondences between HAMSTERS notation elements and error genotypes from the GEMS classification [16]. Such a correspondence is very useful for identifying potential genotypes on an extant task model.

It is important to note that strategic and situational knowledge elements are not present in this table. Indeed, such constructs are similar to the M (Methods) in GOMS and thus correspond to different ways of reaching a goal. As all the methods allow users to reach the goal an error cannot be made at that level and is thus not connected to a genotype.

Table 1. Correspondence between HAMSTERS elements and genotypes from GEMS [16]

Element of notation in HAMSTERS	**Related genotype from GEMS 1616**	
Perceptive task 	Perceptual confusion (Skill Based Error) Interference error (Skill Based Error)	
Input task  Motor task 	Interference error (Skill Based Error) Double capture slip (Skill Based Error) Omissions following interruptions (Skill Based Error)	
Cognitive task 	Skill based errors	Double capture slip Omissions following interruptions Reduced intentionality Interference error Over-attention errors
	Rule based mistakes	Misapplication of good rules First exceptions Countersigns and non-signs Informational overload Rule strength General rules Redundancy Rigidity Application of bad rules Encoding deficiencies Action deficiencies
	Knowledge based mistakes	Selectivity Workspace limitations Out of sight out of mind Confirmation bias Overconfidence Biased reviewing Illusory correlation Halo effects Problems with causality Problems with complexity
Information **Inf :** Information	Double capture slip, Omissions following interruptions, Interference error, all of the Rule Based Mistakes and Knowledge Based Mistakes	
Declarative knowledge **DK :** Declarative	All of the Knowledge Based Mistakes	

New notation elements, based on these correspondences, have been introduced to provide support for identifying and describing human errors in HAMSTERS task models [6].

4 Illustrative example: extract from the 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. This section presents an extract from a case study where the integrated approach has been applied to monitoring and control tasks performed with the Picard ground segment applications.

4.1 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.

Amongst 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.

4.2 Controller's tasks analysis and modelling

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.

The "Start procedure" subroutine is presented in Figure 2. Fine grain modelling 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. The controller can select a procedure from the list and then s/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:

- Select the procedure ("Mouse selection [select procedure]" task, of subroutine type, in Figure 2)
- Start the procedure ("Mouse selection [start procedure] task, of subroutine type, in Figure 2)

Finally, the system will start executing the procedure (system task in Figure 2). This task model also describes which information is required to reach the goal of starting a procedure.

The information about procedure reference. This information is required to be able to search for it in the list and to analyse that the targeted procedure has been found in the list (Box "I: (user) procedure reference with incoming and outgoing

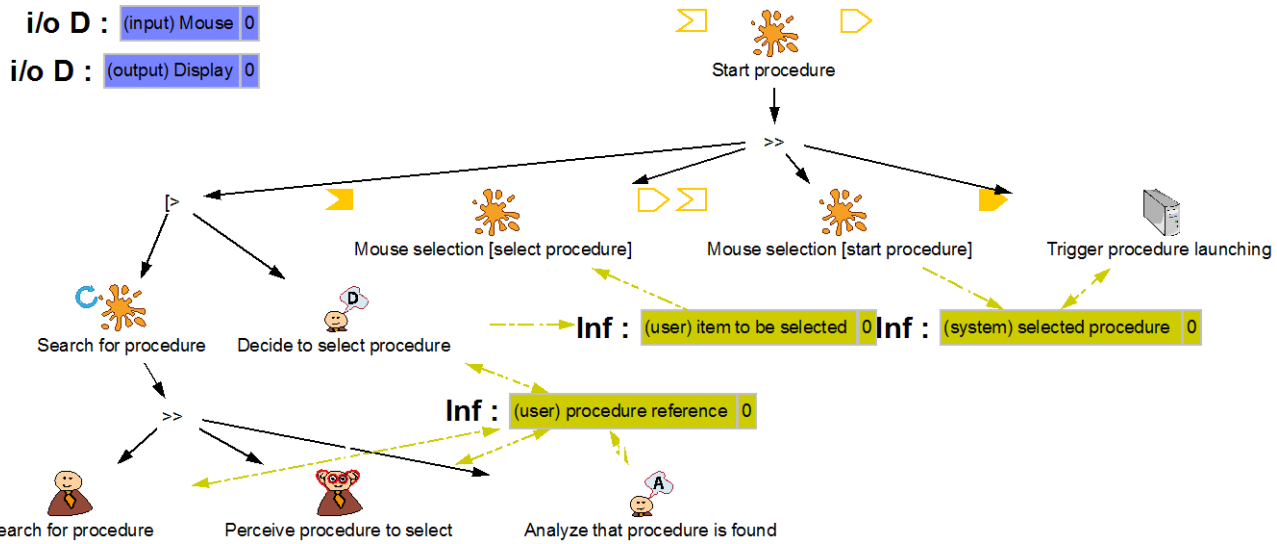

Figure 2. Task model of "Start procedure" task

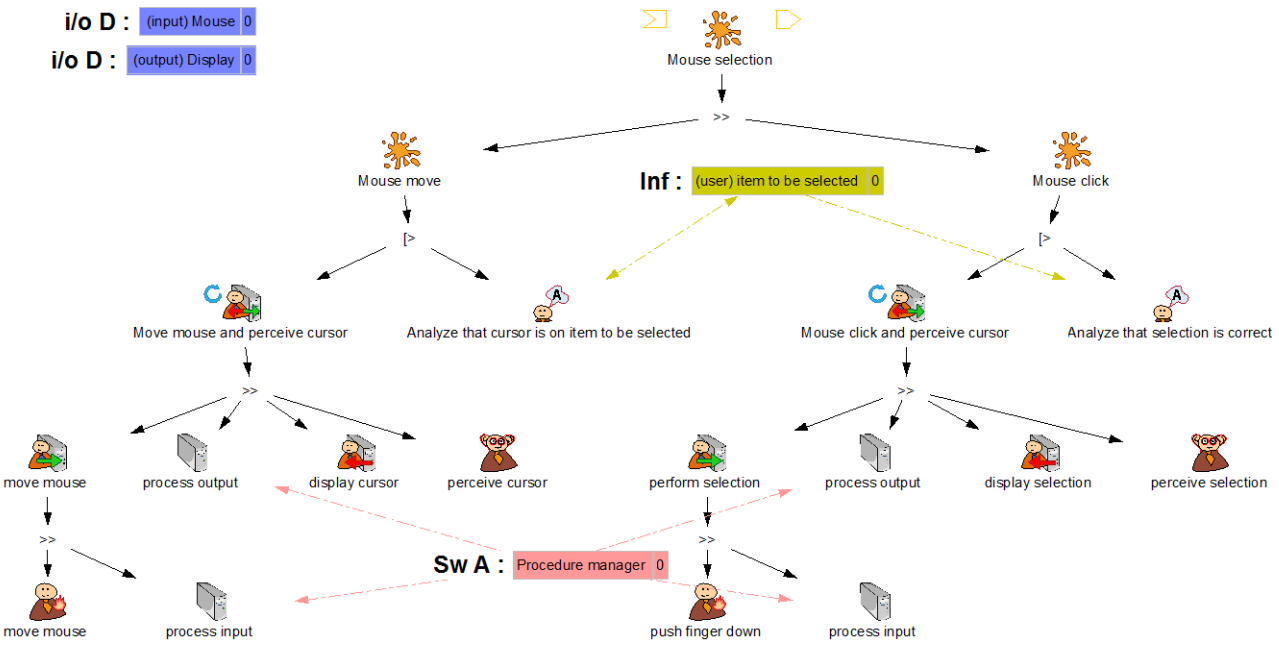

Figure 3. Task model of "Mouse selection" task

arrows to “Search for procedure” user task, “Perceive procedure to select” perceptive task and “Analyze that procedure is found” cognitive analysis task).

The information about the item (of the list) to be selected. Once the controller has decided to select the procedure, s/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). Figure 3 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.

4.3 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 human errors 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. Figure 5 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 depicted in Figure 4). S/he may analyse 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.

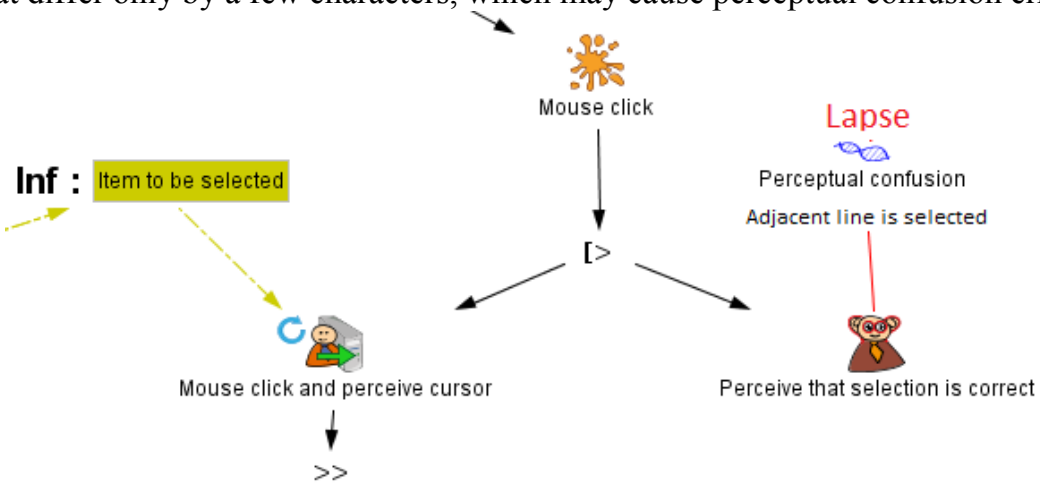

Figure 4. Focus on the action node where an error may occur

#	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
21	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. The unintentionnally selected procedure causes the mission to fail.	Start procedure SWITCH ON SADA2	Push finger down	Other than	Slip	Procedural	Another procedure is selected	Task interrupted	Mission fails.	Catastrophic (4)	Remote (2)	8

Figure 5. Extract from the HEECA table

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 behaviour during the mouse selection task may lead to delay in the mission and be tagged as critical. Figure 5 contains an extract from the HEECA table for the controller's task of driving.

5 Integrating the analysis of organisational faults

Organisational faults usually don't produce immediate faults (i.e. faults that trigger immediate failures and thus interruption of service) but introduce latent faults that under certain conditions and in presence of other faults (either being from the human or system side) [23] may cause failures. They belong to the main source of upstream factors to system (in its broader sense) failures that are leading to incidents and accidents.

STAMP (which stands for Systems Theory Accident Modelling and Process) has been developed by Nancy Leveson [16] and aims at providing a generic framework targeting (among other things) at identifying organisational faults.

One of the main aspects of STAMP is related to the notion of control (see Figure 6) and more precisely in the fact that control is not only based on human and system activities but also on the representations that are constructed by these two actors. For instance the box called Model of Automation inside the Human Supervisor bigger box represents the fact that the behaviour of the operator will be based on the mental representation that he/she owns about the behaviour of the automation currently deployed in the system he/she is supervising. This can explain many sources of accidents/incidents if the constructed mental model is different from the actual behaviour of the system. Such considerations are very similar to the ones that have been driving the field of Human-Computer Interaction over the last 30 years with the action theory from Donald Norman [19]. On the design side of physical or virtual objects the absence (or limited) of discrepancy between these models is named affordance.

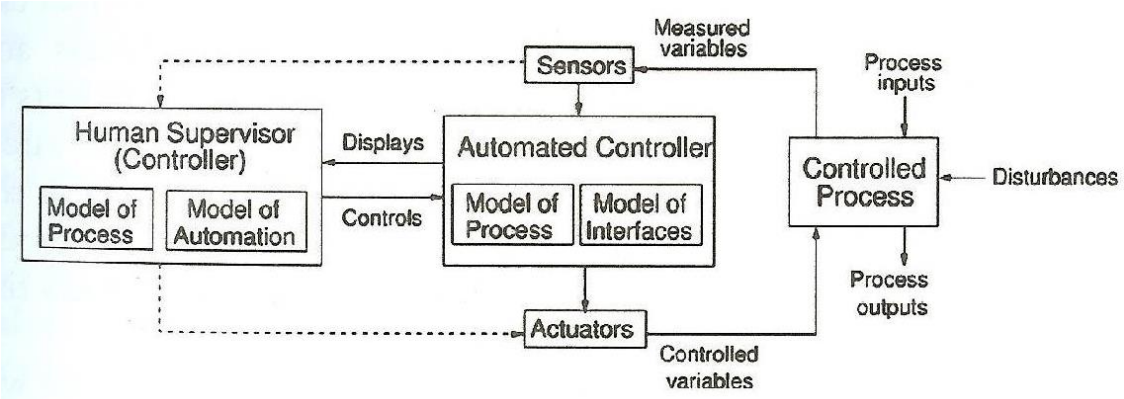

Figure 6. Human and automation roles in control (from [16])

Figure 7 highlights another important aspect of STAMP which is the necessity to take into account both design time and operation time when dealing with socio-technical systems. The detailed process also identifies places where organizational constraints appear and how they can impact system safety and reliability both at design and operation time. STAMP is able to address at a high-level of abstraction a Large Scale Socio Technical System including organization aspects.

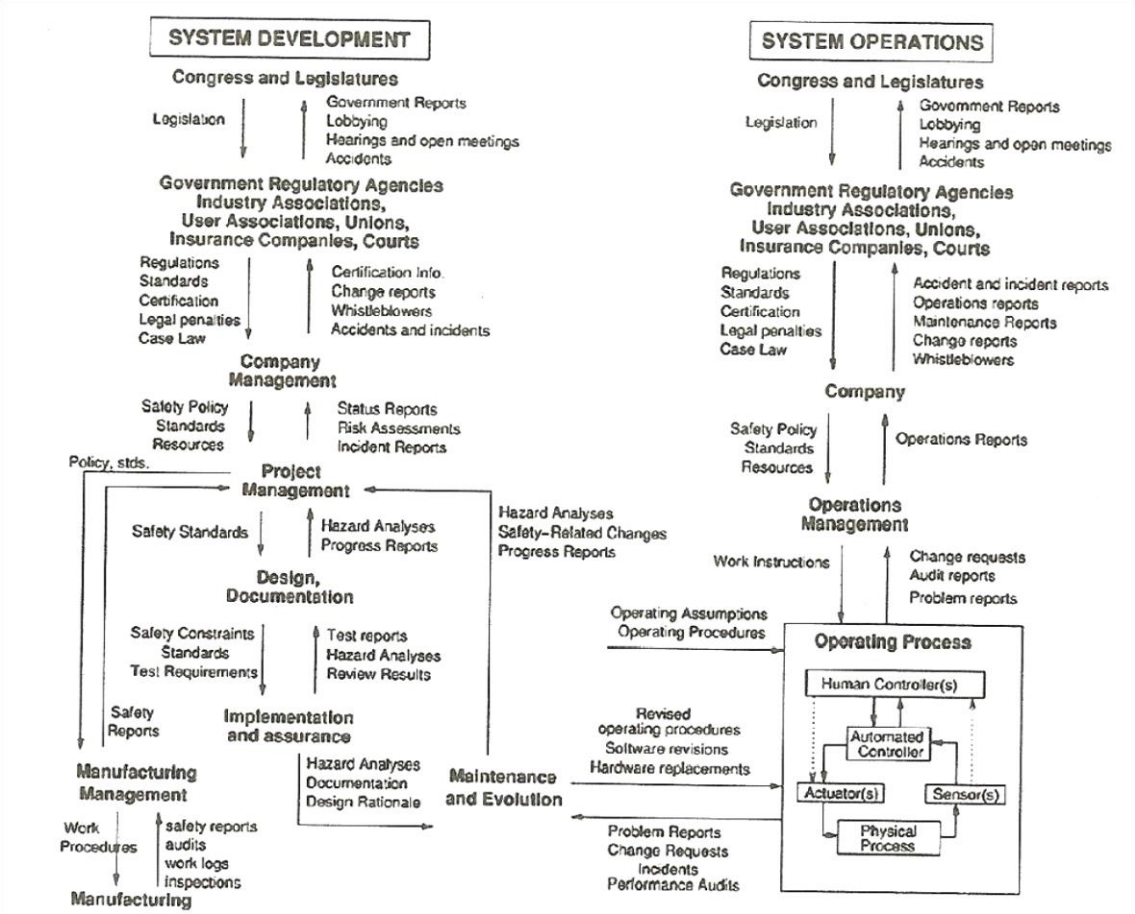

Figure 7. Integrated process dealing with System, Human and Organization views (from [16])

It is important to note that the STAMP analysis only remains at a very high level of abstraction, abstracting away from the details where actual system or human faults occur.

This high-level view will spread throughout the system design as guidelines or choices at design time. For instance the organization might decide to go for quick and cheap development processes that will end up with higher fault rates (especially development faults). Training programs will also influence occurrence of human errors and of operators' capabilities to deal with unexpected infrequent situations requiring a deep understanding of procedures and systems' behaviours that can only be acquired through experience and deep learning.

The control loop explained in the STAMP example provides another perspective based on information flow in the organization that can be made explicit through dedicated techniques such as workflow modelling and analysis [28]. Similar issues arise when modelling multiuser activities (even using HAMSTERS notation). This is where analysis of models can take place to identify (possibly following a high-level approach such as STAMP) the possible missing control and feedback loops.

6 Conclusion

This position paper presented an approach integrating 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 human errors, effects, and criticality is proposed (HEECA). It is inspired and adapted from the FMECA technique.

The key points of the proposed approach are: a) the HEECA technique combining a systematic analysis of the effects of system faults and of human errors, b) a task modelling notation to describe and to assess the impact of system faults and human errors on operators' activities and system performance. These key points have been illustrated on an example extracted from a case study of the space domain that has demonstrated the feasibility of this approach as well as its benefits in terms of identifying opportunities for re-designing the system, re-designing the operations and for modifying operators' training.

Finally, this paper discussed about the main challenges for integrating the analysis of organisation faults in the proposed approach.

References

[1]. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C. Basic concepts and taxonomy of dependable and se-cure computing. In IEEE Trans. on Dependable and Se-cure Computing, vol.1, no.1, pp. 11- 33, Jan.-March 2004.

[2]. Bass, L., et al. (1991). The arch model: Seeheim revisited. In User Interface Developpers' Workshop.

[3]. Bowen, J., & Reeves, S. UI-driven test-first development of interactive systems. In EICS 2011, ACM (2011), 165-174.

[4]. Card, S., Moran, T., Newell, A. The model human pro-cessor: An engineering model of human performance. John Wiley & Sons, 1986.

[5]. Chillarege, R.; Bhandari, I.S.; Chaar, J.K.; Halliday, M.J.; Moebus, D.S.; Ray, B.K.; Wong, M.-Y., "Orthogonal defect classification-a concept for in-process measurements," Software Engineering, IEEE Transactions on , vol.18, no.11, pp.943,956, Nov 1992

[6]. Fahssi, R., Martinie, C., Palanque, P. Enhanced Task Modelling for Systematic Identification and Explicit Representation of Human Errors. In Proc. of IFIP TC 13 Intl. Conf. on HCI, INTERACT 2015, Bamberg.

[7]. Forbrig, P., Martinie, C., Palanque, P., Winckler, M., Fahssi,R. Rapid Task-Models Development Using Sub-models, Sub-routines and Generic Components. Proc.of HCSE 2014, pp. 144-163.

[8]. Hecht H. and Fiorentino E. Reliability assessment of spacecraft electronics. In Annual Reliability and Maintainability Symp., pages 341–346. IEEE, 1987.

[9]. Hollnagel, E. Cognitive reliability and error analysis method (CREAM). Elsevier, 1998.

[10]. Martinie, C., Barboni, E., Navarre, D., Palanque, P., Fahssi, R., Poupart, E., Cubero-Castan, E. Multi-models-based engineering of collaborative systems: application to collision avoidance operations for spacecraft. Proc. of EICS 2014, pp. 85-94.

[11]. Martinie, C., Palanque, P., & Winckler, M. (2011). Structuring and composition mechanisms to address scalability issues in task models. In Human-Computer Interaction–INTERACT 2011 (pp. 589-609). Springer Berlin Heidelberg.

[12]. Martinie, C., Palanque, P., Barboni, E., & Ragosta, M. (2011, October). Task-model based assessment of automation levels: application to space ground segments. In Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on (pp. 3267-3273). IEEE.

[13]. Martinie, C., Palanque, P., Fahssi, R., Blanquart, J.-P., Fayollas, C., Seguin, C. Task Model-Based Systematic Analysis of Both System Failures and Human Errors. IEEE Transactions on Human-Machine Systems, to appear in 2015.

[14]. Martinie, C., Palanque, P., Fahssi, R., Blanquart, J-P., Fayollas, C., Seguin, C. Task Models Based Systematic Analysis of both System Failures and Human Errors. IEEE Transactions on Human Machine Systems, special issue on Systematic Approaches to Human-Machine Interface: Improving Resilience, Robustness, and Stability, to appear.

[15]. Martinie, C., Palanque, P., Ragosta, M., & Fahssi, R. (2013, August). Extending procedural task models by systematic explicit integration of

objects, knowledge and information. In Proceedings of the 31st European Conference on Cognitive Ergonomics (p. 23). ACM.

[16].Leveson N. A Systems-Theoretic Approach to Safety in Software-Intensive Systems. IEEE Trans. Dependable Sec. Comput. 1(1): 66-86 (2004).

[17].Navarre, D., Palanque, P., Ladry, J. F., & Barboni, E. (2009). ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM TOCHI, 16(4), 18.

[18].Nicolescu B., Peronnard P., Velazco R., and Savaria Y. Efficiency of Transient Bit-Flips Detection by Software Means: A Complete Study. Proc. of the 18th IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems (DFT '03). IEEE Computer Society, 377-384.

[19].Norman, D. The Design of Everyday Things (Originally published: The psychology of everyday things). New York: Basic Books, 1988.

[20].Norman, D. A. (1981). Categorization of action slips. Psychological review, 88(1), 1.

[21].Rasmussen, J. Skills, rules, knowledge: signals, signs and symbols and other distinctions in human performance models, IEEE transactions: Systems, Man &Cybernetics, 1983.

[22].Reason J. Generic error modelling system: a cognitive framework for locating common human error forms. New technology and human error, 63, 86. 1987.

[23].Reason J. Managing the Risks of Organizational Accidents. Ashgate Publishing Limited, 1997.

[24].Reason, J. Human Error, Cambridge University Press. 1990

[25].Regis, D.; Hubert, G.; Bayle, F.; Gatti, M., "IC components reliability concerns for avionics end-users," Digital Avionics Systems Conference IEEE/AIAA 32nd pp.2C2-1,2C2-9, 5-10 Oct. 2013.

[26].Ruksenas R., Curzon P., Blandford A., Back J. Combining Human Error Verification and Timing Analysis. EHCI/DS-VIS 2007: 18-35

[27].Schroeder B., Pinheiro E., and Weber W.-D.. DRAM errors in the wild: a large-scale field study. In ACM SIGMETRICS, pages 193–204, Seattle, WA, June 2009.

[28].van der Aalst W., ter Hofstede A. YAWL: yet another workflow language. Inf. Syst. 30(4): 245-275 (2005)