Beyond Formal Methods for Critical Interactive Systems: Dealing with Faults at Runtime

Camille Fayollas[2,3], **Célia Martinie**[2], **Philippe Palanque**[2], **Yannick Deleris**[1]

[1] AIRBUS Operations,
316 Route de Bayonne,
31060, Toulouse, France
yannick.deleris@airbus.com

[2]ICS-IRIT, University of Toulouse,
118 Route de Narbonne,
F-31062, Toulouse, France
{Name}@ irit.fr

[3]LAAS-CNRS,
7 avenue du colonel Roche,
F-31077 Toulouse, France
cfayolla@laas.fr

ABSTRACT
Formal methods provide support for validation and verification of interactive systems by means of complete and unambiguous description of the envisioned system. They can also be used (for instance in the requirements/needs identification phase) to define precisely what the system should do and how it should meet user needs. If the entire development process in supported by formal methods (for instance as required by DO 178C [7] and its supplement 333 [8]) then classical formal method engineers would argue that the resulting software is defect free. However, events that are beyond the envelope of the specification may occur and trigger unexpected behaviors from the formally specified system resulting in failures. Sources of such failures can be permanent or transient hardware failures, due to (when such systems are deployed in the high atmosphere e.g. aircrafts or spacecrafts) natural faults triggered by alpha-particles from radioactive contaminants in the chips or neutron from cosmic radiation. This position paper proposes a complementary view to formal approaches first by presenting an overview of causes of unexpected events on the system side as well as on the human side and then by discussing approaches that could provide support for taking into account system faults and human errors at design time.

Author Keywords
Formal methods; interactive systems; human reliability; fault-tolerant systems.

ACM Classification Keywords
D.2.2 [Software] Design Tools and Techniques – Computer aided software engineering (CASE).

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.

Building reliable interactive systems is a cumbersome task due to their very specific nature. The behavior of these reactive systems is event-driven. As these events are triggered by human operators, these systems have to react to unexpected events. On the output side, information (such as the current state of the system) has to be presented to the operator in such a way that it can be perceived and interpreted correctly. Lastly, interactive systems require addressing together hardware and software aspects (e.g. input and output devices together with their device drivers).

In the dependable computing domain, empirical studies have demonstrated (e.g. [20]) that software failures may occur even though the development of the system has been extremely rigorous. One of the many sources of such failures 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 [31] concerns systems deployed in the high atmosphere (e.g. aircrafts) or in space (e.g. manned spacecraft [13]). 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.

WHY FORMAL METHODS AND ZERO DEFECT APPROACHES ARE NOT ENOUGH
To be able to ensure that the system will behave properly whatever happens, a system designer has to consider all the issues that can impair the functioning of that system. In the perspective of identifying all of them, in the domain of dependable computing, Avizienis et al [1] have defined a typology of faults. This typology leads to the identification of 31 elementary classes of faults. Figure 1 presents a simplified view of this typology and makes explicit the two main categories of faults (top level of the figure): i) the ones occuring at development time (including bad designs, programming errors, …) and ii) the one occuring at opera-

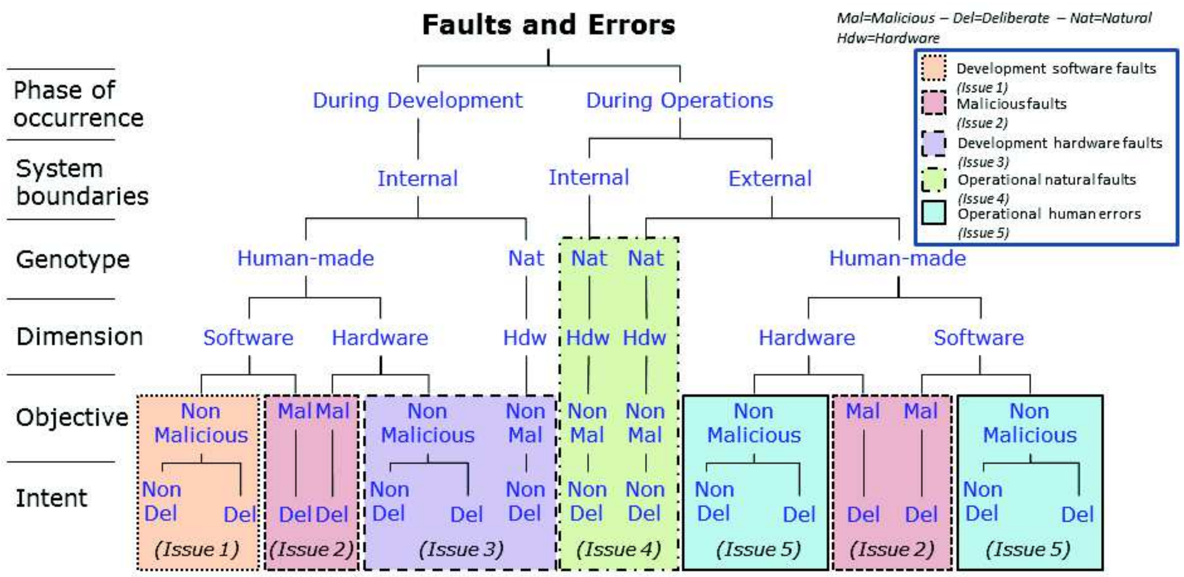

Figure 1. Typology of faults in computing systems (adapted from [1]) and associated issues for the resilience of these systems

tion times (right-hand side of the figure including user error such as slips, lapses and mistakes as defined in [25]).

We propose to organize the leaves of the typology in five different groups as each of them brings a special problem (issue) to be addressed:

- *Development software faults (issue 1)*: software faults introduced by a human during the system development.
- *Malicious faults (issue 2)*: faults introduced by human with the deliberate objective of damaging the system (e.g. causing service denial or crash of the system).
- *Development hardware faults (issue 3)*: natural (e.g. caused by a natural phenomenon without human involvement) and human-made faults affecting the hardware during its development.
- *Operational natural faults (issue 4)*: faults caused by a natural phenomenon without human participation, affecting the hardware and occurring during the service of the system. As they affect hardware, they are likely to damage software as well.
- *Operational human-errors (issue 5)*: faults resulting from human action during the use of the system. These faults are particularly of interest for interactive system and the next subsection describe them in detail.

We consider that malicious faults are beyond the scope of this position paper and will thus not be further discussed. However, it might be interesting within the workshop to address this aspect that is more and more relevant with the open, collaborative interactive systems.

Considering system faults

In the domain of fault-tolerant systems, empirical studies have demonstrated (e.g. [20]) 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 [31] concerns systems deployed in the high atmosphere (e.g. aircrafts) or in space (e.g. manned spacecraft [13]). Furthermore the evolution of modern IC components may lead in the next future to a higher probability of physical faults in operation. Although the recommendation for avionics systems is 100 FITs over 25 years lifetime, the current Deep Sub-Micron (DSP) technology may lead to a failure rate up to 1000 FITs, only during 5 years operational life time [28]. 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.

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 analyze human error. Norman, proposed a predictive model for errors [21], 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) [25]. This model provides support for reasoning about possible human errors and has been used to classify error types. Reason [26] 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 [21]; 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 [26] (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 [5].

Causes of errors and their observation are different concepts that should be separated when analyzing user errors. To do so, Hollnagel [15] 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 [29]. As pointed out in [21] 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.

PROPOSALS FOR DEALING WITH SYSTEM FAILURES AND HUMAN ERRORS

Although system failures and human errors can both occur at runtime and be strongly correlated, these two problems are handled separately when developing an interactive system.

Dealing with operational natural faults

The issue of operational natural faults has hardly been studied in the field of human-computer interaction and just a few contributions are available about this topic. However, this issue has long been studied in the field of dependable computing systems. As the operational natural faults are unpredictable and unavoidable, the dedicated approach for dealing with them is fault-tolerance [1] that can be achieved through specialized fault-tolerant architectures, 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 [19].

To deal with these faults, we proposed two approaches:

- The reconfiguration of the interaction techniques or possibly the organization of display when required by the occurrence of hardware faults [18].
- The adaptation of fault-tolerant architecture for developing fault-tolerant widgets as proposed in [33] or for extending this approach to all the interactive components of the interactive system (including for example the interaction techniques) as proposed in [10].

Dealing with human errors

Many techniques have been proposed for identifying which human errors may occur in a particular context and what could be their consequences in this given context.

- Several human reliability assessment techniques such as CREAM [12], HEART [35], and THERP [33] are based on task analysis. They provide support to assess the possibility of occurrence of human errors by structuring the analysis around task descriptions. Beyond these commonalities, THERP technique also provides support for assessing the probability of occurrence of human errors.
- Task models based techniques have also been proposed to identify, describe and analyze potential human errors and human tasks deviation such as in [29], [9] and [23].

Dealing with both operational natural faults and human errors

Integrated approaches can be envisioned for taking into account both system faults and human errors. Such approaches can leverage existing techniques in the fields of: dependable computing, human reliability assessment and human computer interaction. As proposed in [16], a stepwise and iterative process can be used to identify in a systematic way human error and system failures for an under-development interactive systems. From this systematic identification, the construction of enriched task models (embedding potential human errors and system faults), can provide support for analyzing their impact and proposing changes for modifying the system.

ILLUSTRATIVE EXAMPLE FROM THE ATM WORLD

The typology of faults introduced in Figure 1 can be easily applied to any application providing support to understanding how the approach followed for the development of a system is addressing the various faults.

In the case of AMAN application proposed for the workshop, the various faults can lead to failures in the management of the aircrafts by the air traffic controllers. For instance, as detailed in [17], we have analysed 3 types of failures leading to 3 automation degradation scenarios: advisories from AMAN being not available anymore, advisories being frozen for a while then starting again and advisories provided being delayed.

If a rigorous development process is followed and formal methods are used (as proposed in DO178-C [7]) one could expect that such failures would not occur. However, natural faults could easily produce such undesired behaviours. Similarly human errors such as not perceiving the advisories or interpreting them incorrectly could also end up with similar malfunction (but this time at organizational level only as the system is supposed to function correctly).

CONCLUSION

This position paper argues that formal methods are good candidates for dealing with development faults. However, this position paper has also presented a typology of faults that identify other sources of failures that development faults: natural faults and human errors.

In order to cover all these faults and to prevent related failures to occur we argued that multiple combined approaches (including formal methods) should be applied. For instance, it is interesting to note that detection and recovering mechanisms for natural faults could be described using formal methods in order to guarantee that their behaviour will be conformant with the expected one (as presented in [34]).

We have not addressed issues related to malicious faults that could however be discussed during the workshop.

REFERENCES

1. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. In IEEE Trans. on Dependable and Secure Computing, vol.1, no.1, pp. 11- 33, Jan.-March 2004.
2. Back J., Blandford A, CurzonP. Recognising Erroneous and Exploratory Interactions. INTERACT (2) 2007: 127-140
3. Barboni, E., Bastide, R., Lacaze, X. Navarre, D., Palanque, P. Petri Net Centered versus User Centered Petri Nets Tools. AWPN 2003 - 10th Workshop on Algorithms and Tools for Petri Nets, Eichstätt, Germany, 26/09/2003-27/09/2003.
4. Beaudouin-Lafon, M. et al. CPN/Tools: A Post-WIMP Interface for Editing and Simulating Coloured Petri Nets. In Proc. of ICATPN'2001: 22nd International Conference on Application and Theory of Petri Nets (June 2001, Newcastle upon Tyne, England), Lecture Notes in Computer Science, Springer-Verlag, 2001, pp. 71-80.
5. Card, S., Moran, T., Newell, A. The model human processor: An engineering model of human performance. John Wiley & Sons, 1986.
6. Dix, A. Upside down As and algorithms – computational formalisms and theory. J. Carroll (Ed.), HCI Models Theories and Frameworks: Toward a Multidisciplinary Science, Morgan Kaufmann, San Francisco (2003), pp. 381–429 (Chapter 14).
7. DO-178C / ED-12C, Software Considerations in Airborne Systems and Equipment Certification, published by RTCA and EUROCAE, 2012
8. DO-333 Formal Methods Supplement to DO-178C and DO-278ASoftware Tool Qualification Considerations, published by RTCA and EUROCAE December 13, 2011
9. 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.
10. Fayollas C., Fabre J-C., Palanque P., Barboni E., Navarre D., Deleris Y: Interactive cockpits as critical applications: a model-based and a fault-tolerant approach. Int. Journal on Critical Component-Based Software 4(3): 202-226 (2013)
11. Fayollas, C., et al. A Software-Implemented Fault-Tolerance Approach for Control and Display Systems in Avionics. In Proc. PRDC 2014, IEEE, 21-30.
12. Gram, C., Cockton, G. Design principles for Interactive Software. London. 1996. Chapman & Hall.
13. Hecht H. and Fiorentino E. Reliability assessment of spacecraft electronics. In Annual Reliability and Maintainability Symp., pages 341–346. IEEE, 1987.
14. Hollnagel E. “The phenotype of erroneous actions Implications for HCI design,” in G. R. S. Weir and J. L. Alty, Eds , Human Computer Interaction and the Complex Systems, London: Academic Press, 1991.
15. Hollnagel, Erik. Cognitive reliability and error analysis method (CREAM). Elsevier, 1998.
16. Martinie C., Palanque P., Ragosta M., Sujan M-A., Navarre D., Pasquini A.: Understanding Functional Resonance through a Federation of Models: Preliminary Findings of an Avionics Case Study. SAFECOMP 2013: 216-227
17. 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.
18. Navarre, D., Palanque, P. 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.

19. Neema, S., Bapty, T., Shetty, S., and Nordstrom; S. Autonomic fault mitigation in embedded systems. Eng. Appl. Artif. Intell., vol. 17, no. 7,pp. 711–725, 2004.
20. 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.
21. Norman, D. A. (1981). Categorization of action slips. Psychological review, 88(1), 1.
22. Papatzanis, G, Curzon, P., Blandford, A. Identifying Phenotypes and Genotypes: A Case Study Evaluating an In-Car Navigation System. EHCI/DS-VIS 2007, pp. 227-242.
23. Paterno, F., Santoro, C., "Preventing user errors by systematic analysis of deviations from the system task model", 2002, *Int. Journal on Human Computing Systems*, Elsevier, vol. 56, n. 2,pp. 225-245.
24. Polet, P, Vanderhaegen, F, and Wieringa, P. Theory of safety related violation of system barriers. Cognition Technology & Work, 4, 3, 171-179. 2002.
25. Rasmussen, J. Skills, rules, knowledge: signals, signs and symbols and other distinctions in human performance models, IEEE transactions: Systems, Man &Cybernetics, 1983.
26. Reason, J T. Generic error modelling system: a cognitive framework for locating common human error forms. New technology and human error, 63, 86. 1987.
27. Reason, J. (1990). Human Error, Cambridge University Press
28. 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
29. Rimvydas Ruksenas, Paul Curzon, Ann Blandford, Jonathan Back: Combining Human Error Verification and Timing Analysis. EHCI/DS-VIS 2007: 18-35
30. Rizzo, A., Ferrante, D., & Bagnara, S. (1995). Handling human error. Expertise and technology: Cognition & human-computer cooperation, 195-212.
31. Schroeder B., E. Pinheiro, and W.-D. Weber. DRAM errors in the wild: a large-scale field study. In ACM SIGMETRICS, pages 193–204, Seattle, WA, June 2009.
32. Silva, J-L., Fayollas, C., Hamon, A., Palanque, P., Martinie, C., Barboni, E. Analysis of WIMP and Post WIMP Interactive Systems based on Formal Specification. International Workshop on Formal Methods for Interactive Systems (FMIS 2013), London, 24/06/2013, Electronic Communications of the EASST.
33. Swain, A. D., Guttman, H. E. Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications, Final report. NUREG/CR- 1278. SAND80-0200. RX, AN. US Nuclear Regulatory Commission, August 1983.
34. Tankeu-Choitat, A. et al. Self-checking components for dependable interactive cockpits using formal description techniques. In Proc. PRDC 2011, 164-173.
35. Williams, J. C. A data-based method for assessing and reducing human error to improve operational performance. In Human Factors and Power Plants, IEEE, 1988. p. 436-450.