# Interactive Systems: a Unique Place for Human-Hardware-Software Integration and their Vulnerability to Human-Made and Natural Faults

Célia Martinie, Philippe Palanque


# Interactive Systems: a Unique Place for Human-Hardware-Software Integration and their Vulnerability to Human-Made and Natural Faults

Philippe Palanque
ICS-IRIT,
Université Paul Sabatier –Toulouse III
118, route de Narbonne, Toulouse, France
+33 561 55 6965
palanque@irit.fr

Célia Martinie
ICS-IRIT,
Université Paul Sabatier –Toulouse III
118, route de Narbonne, Toulouse, France
+33 561 55 6965
martinie@irit.fr



**Abstract**. Interactive systems are complex systems that allow operators to control and monitor other systems. The interactive systems offer one or several user interfaces, composed of hardware and software components, which are used by the operators to perform their tasks. The complexity of interactive systems lays in its nature (hardware and software integration) but also in the rapidly evolving technology (new input and output devices, new interaction techniques …). This paper argues that these interactive systems are the very place where human-system integration takes place. It is thus important to design, develop, test, certify and deploy them very carefully. Unfortunately, methods, techniques and tools from software and system engineering need deep tuning to be adapted to their characteristics. The paper presents a generic, customizable interactive system architecture and highlights how its components relate to the human operator. It also presents a taxonomy of possible faults that may degrade the behavior of the various components. We demonstrate that these two contributions can be jointly used to systematically identify possible faults (both in the operator and in the system) and proposes mechanisms to prevent, remove or tolerate them.

## Introduction

Interactive systems are complex systems that allow operators to control and monitor other systems. The interactive systems offer one or several user interfaces, composed of hardware and software components, which are used by the operators to perform their tasks. The complexity of interactive systems lays in its nature (hardware and software integration) but also in the rapidly evolving technology (new input and output devices, new interaction techniques …).

The diversification of technological platforms on which interactive systems are designed, developed and deployed significantly increases the complexity of designers and developers' tasks. At the same time, such an ever-changing context has made it very difficult for researchers belonging to the engineering community on interactive systems, to provide generic approaches to support those tasks. Designers need to go beyond the interactive application design by providing new interaction techniques that encompass new input and output devices, which can be very cumbersome to design and evaluate (as, for instance, multimodal interactions involving multi-touch gestures (Hamon et al. 2013)). Developers of these systems are repetitively facing the same issues of: i) new devices integration, software redesign (due to device drivers' evolution) and above all poor reliability of the

resulting system due to the low level of maturity of the various components to integrate. Such constraints are even stronger in the area of critical systems where a failure may lead to catastrophic consequences

This paper argues that these interactive systems are the very place where human-system integration can take place. It is thus important to design, develop, test, certify and deploy them very carefully. Unfortunately, methods, techniques and tools from software and system engineering need deep tuning to be adapted to their characteristics. The paper presents a generic tunable interactive system architecture that exhibits its software and hardware components but also how these components relate to the human operator. We demonstrate that this architecture can be also used in order to systematically identify possible faults (both in the operator and in the system) and proposes mechanisms to prevent, remove or tolerate them.

Next section presents a generic tunable hardware and software architecture encompassing the human operator. The following section presents an integrated taxonomy of faults that can occur in the system as well as in the operator. We then present concrete examples of such faults and where these faults are located on the generic architecture. We then highlight some known solutions to address these faults as well as open challenges. The last section concludes the paper.

## Architectural view on Human Hardware Software Integration in an Interactive System

Figure 1 presents an architectural view (from left to right) of the operator, the interactive command and control system, and the underlying cyber-physical system (e.g., an aircraft engine). This architecture is a simplified version of MIODMIT (Multiple Input and Out-put Devices and Multiple Interaction Techniques), a generic architecture for multi-modal interactive systems (Cronel et al. 2019) highlighting functions (rounded box) and information flow (arrows). The top part of the figure differentiate the hardware and software components of an interactive system. Interaction with the operator takes place only through manipulation of hardware input devices and perception of information from hardware output devices (which includes the cyber-physical system).

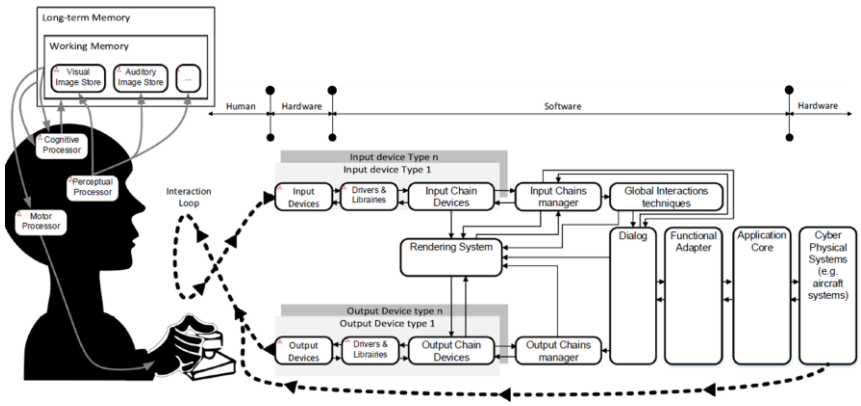


Figure 1. H-MIODMIT architecture (adapted from (Cronel et al. 2019))

**The System side.** The right side of the Software section of the architecture (starting with Dialog box) corresponds to what is usually called interactive applications. This is where HCI methods such as task analysis are needed for building usable application that fit the operators' work (Diaper and Stanton 2003). In the context of command and control systems, this interactive application is connected to the so-called Cyber-Physical System to operate (which is also most of the time composed of hardware and software components).

**The Human side.** The left side of Figure 1 represents the operator's side. The drawing is based on work that models the human as an *information processor* (Card et al 1983), based on previous research in psychology. In that model, the human is presented as a system composed of three interconnected processors. The *perceptive system* senses information from the environment – primarily the visual, auditory and tactile systems as these are the most commonly used for interacting with computers. The *motor system* allows operators to act on the real world. Target selection (a key interaction mechanism) has been deeply studied (Soukoreff & MacKenzie 2004); for example, Fitts' Law provides a formula for predicting the difficulty for an operator to select a target, based on its size and distance (Fitts 1954). The *cognitive system* is in charge of processing information gathered by the perceptual system, storing that information in memory, *analyzing* the information and *deciding* on actions performed using the motor system. The sequential (sometimes parallel) use of these systems (perceptive, cognitive and motoric) while interacting with computers is called the Human-Computer Interaction Loop (HCIL). The operator's *memory* (decomposed into short term and long-term memory) is known for being subject to alteration and decay (Murre & Dross 2015).

**The specificities of the Interaction.** The top left of the Software section corresponds to the interaction technique that uses information from the input devices. Interaction techniques have a tremendous impact on operator performance. Standard interaction techniques encompass complex mechanisms (e.g. modification of the cursor's movement on the screen according to the acceleration of the physical mouse on the desk). This design space is of prime importance and HCI research has explored multiple possibilities for improving performance, such as enlarging the target area for selection on touch screens (Olwal & Feiner 2003) and providing on-screen widgets to facilitate selection (Albinsson, P.A. & Zhai 2003).

The interaction mainly takes place though the manipulation of input devices (e.g., keyboard or mouse) and the perception of information from the output devices (e.g., a computer screen or speaker). Another channel usually overlooked is the direct perception by the operator of information produced (usually as a side effect and not on purpose) of the underlying cyber-physical systems (e.g., noise or vibrations from an aircraft engine (represented by the dotted line at the bottom of Figure 1)).

## Integrated classification of faults that may occur in interactive systems

(Avizienis et al. 2004) proposed a classification of faults according to their root cause (phenomenological cause also called genotype), the part they affect (dimension) and the phase of the life cycle when then may occur (phase). Faults may be caused by human actions or by natural phenomena without human participation ("*due to natural processes that cause physical deterioration*"). They may affect the software part of the system (SW) or the hardware part of the system (HW). They may occur during the development of the system or during operations, when users use it to perform their tasks (top of the taxonomy on Figure 2).

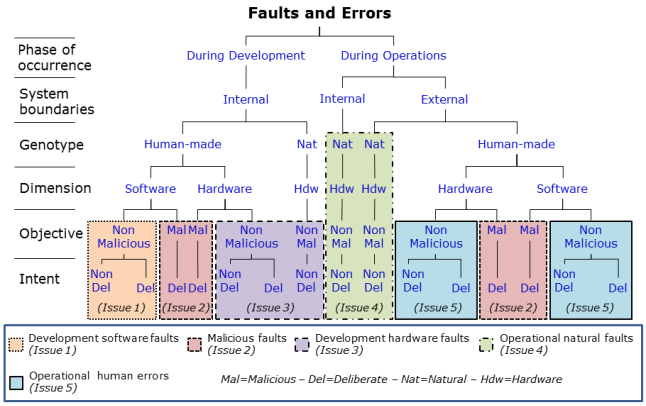


Figure 2. Taxonomy of faults that may affect the behavior of the computing systems (adapted from (Avizienis et al. 2004)

More recently, (Palanque et al. 2020) proposed a classification of faults affecting the behavior of the operator. This classification follows the same decomposition pattern as the one from (Avizienis et al. 2004) but adapts it to the inner behavior of operators. First, it extends the *System boundary* dimension to separate human faults from external causes. Second, it adds new levels to the *Phenomenological cause* dimension to distinguish between faults arising 1) from the operator, 2) from another person, and 3) from the natural world (including the system itself). Third, it introduces the *Human capability* dimension to differentiate faults in the operator's perceptual, cognitive, and motor abilities. Fourth, it adds specific fault categories that derive from these dimensions usually not accounted for such as cognitive biases, for which a collection can be found in (Benson 2016).

As the Human Computer Interaction Loop (HCIL) is the location of integration between operators and computing systems there is a need to integrate both classifications to identify, in a single framework, faults induced in the system and faults induced in the operator. Figure 3 presents such an integrated view on both classifications.

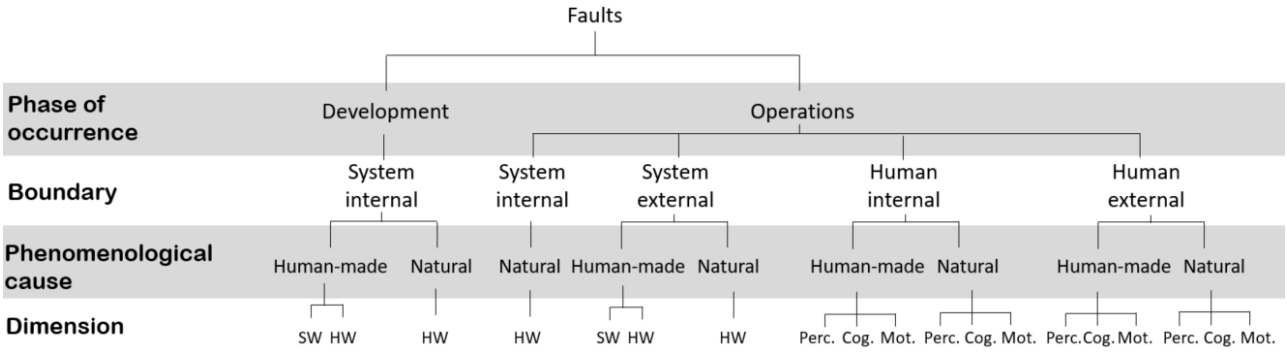


Figure 3. Integration of the HCI Loop classification (Palanque et al. 2020) within the Avizienis et al. fault classification (Avizienis et al. 2004)

Both classifications enable to characterize a fault according to its objective (malicious or non-malicious) and intent (deliberate or non-deliberate). In the case where the objective is malicious, the human deliberately causes the fault. Natural faults are non-malicious and non-deliberate. Due to space constraints, this paper do not present this refinement of the classifications in the presented integrated view.

## Illustrative examples within a commercial aircraft cockpit

We propose to use the H-MIODMIT architecture and the integrated view on fault classifications to identify exhaustively the faults that may possibly occur at development time and at operation time and this for the whole Human Computer Interaction Loop. To illustrate this proposal, we present nine illustrative examples of concrete possible faults that may occur in an aircraft cockpit. Each of them belongs to a main branch of the integrated view on classifications (highlighted in Figure 4), and each of them affects a part of the H-MIODMIT architecture (highlighted in Figure 5).

In the following paragraphs, we use a common description template for presenting the examples. First, we introduce the example of fault, its dimension (affected part according to the classification) and the corresponding affected part in the H-MIODMIT architecture. We then present the main additional characteristics of the presented fault: boundary, phenomenological cause and phase of the system life cycle when it may occur. In the case where the example of fault is specifically malicious and/or deliberate, we explicitly indicate it. At last, we describe the possible failures that may occur because of this fault, as well as the possible ways of dealing with this fault.

Fault labelled 1 in Figure 5 is a warning sound missing some music notes. This is a software fault that affects the rendering function (audio). It is internal to the system and a developer may cause it during development, by not writing a line of code that was supposed to command the production of

a subset of notes of the warning sound. It is possible to avoid such fault by applying fault removal techniques at development time, such as peer review of the software and code inspection meetings.

Fault labelled 2 in Figure 5 is a weak button. This is a hardware fault that affects an input device. It is internal to the system and a natural phenomenon may cause it during development. For example, in the factory, due to a micro earthquake, a production bench may malfunction and release a button which material is weak and more fragile than expected. Such a button may break before planned maintenance in the cockpit. It is possible to remove this fault (if it is known) by testing buttons and removing the weak ones.

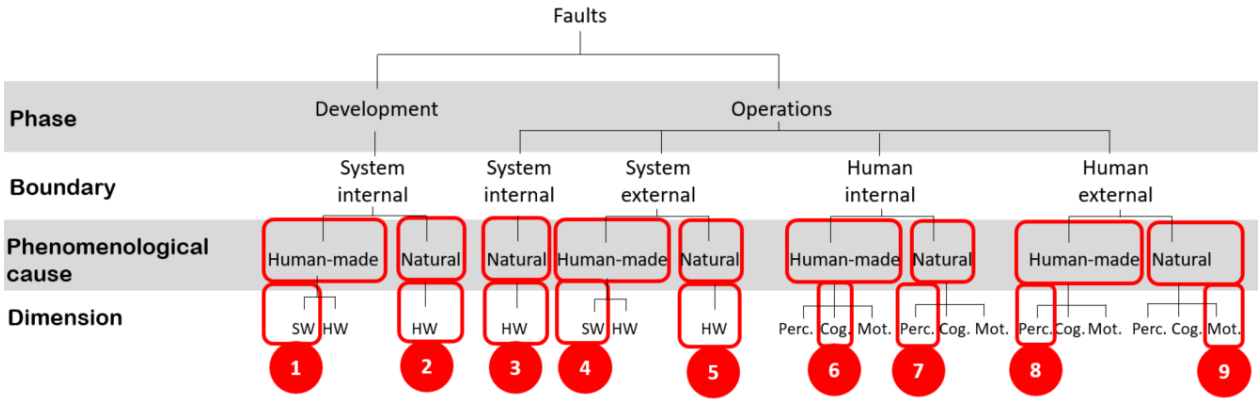


Figure 4. Integrated classification with references to examples of faults highlighted in each branch

Fault labelled 3 in Figure 5 is a broken button. This is a hardware fault that affects an input device. It is internal to the system and a natural phenomenon may cause it during operations. A broken button may prevent the crewmembers to trigger a command. It is possible to remove this fault by planning regular maintenance activities during operations.

Fault labelled 4 in Figure 5 is the weather radar application being in an unexpected state. This is a software fault that affects the dialog of the weather radar application. It is external to the system and human-made during operations. It may occur if the pilot inputs an invalid tilt angle value for the configuration of the weather radar. Such invalid value may put the weather radar in an unexpected state and freeze the dialog. It is possible to remove such fault by applying verification technique at development time to check that whatever the input made by the operator, the data will not put the application dialog in an unexpected state.

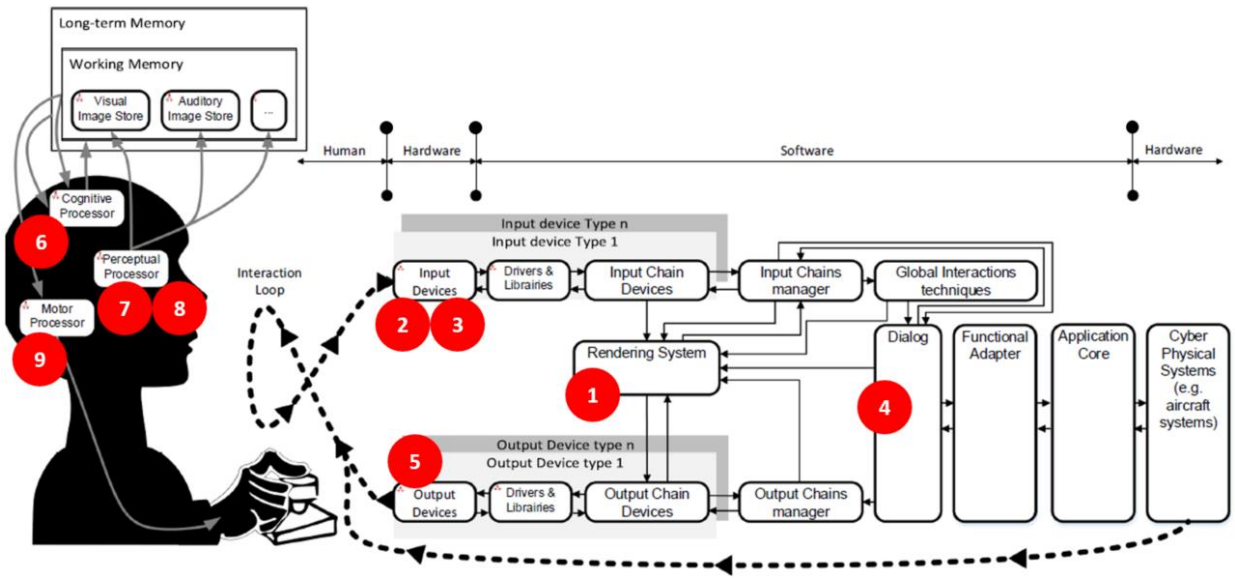


Figure 5. H-MIODMITT architecture with references to examples of faults

Fault labelled 5 in Figure 5 is a bit flip in a register of the graphical processing unit of the system display (in charge of presenting the information about the aircraft systems). This is a hardware fault that affects an output device. It is external to the system and a natural phenomenon may cause it during operations. External radiations may modify the content of one of the registers of the graphical processing unit of the system display. Such modification may trigger the display of an inaccurate value of the current pressure in the cabin, which may cause the crewmembers to take inappropriate actions to reach a target cabin pressure. It is possible to cope with such fault by applying redundancy mechanisms to present the information on two different output devices.

Fault labelled 6 in Figure 5 is the anchoring effect of a crew member focusing on a subset of situational information. This is a human fault that affects the cognitive processor. It is internal to the human and may happen during operations. A crewmember who focuses on the first perceived warning information and do not notice incoming new information may make a wrong analysis of the situation, which may lead the crew to apply inappropriate actions. This cognitive fault is a bias referenced as the anchoring effect (Benson 2016). It is possible to cope with such fault by making the displays stop presenting information for a while if the crewmember did not acknowledged new incoming information.

Fault labelled 7 in Figure 5 is the reduced sight of a crew member. This is a fault affecting the human in the perceptual processor. It is internal to the human and may happen during operations. Because of aging, a crewmember may not able to see correctly the displayed information. It is possible to cope with such fault by assigning recurrent medical examination to crewmembers in order to ensure that their vision is accurate with respect to their missions.

Fault labelled 8 in Figure 5 is the temporary blocked sight of a crew member. This is a human fault that affects the perceptual processor. It is external to the human and may happen during operations. A crewmember may be not able to perceive visually the cockpit displays and controls, at least partially, if this crewmember is victim of a laser attack from a person on the ground (malicious and deliberate in the case of this example). It is possible to cope with such fault by equipping crewmembers with laser proof goggles.

Fault labelled 9 in Figure 5 is the involuntary movement of a crew member. This is a human fault that affects the motor processor. It is external to the human and may happen during operations. Vibrations may occur suddenly while flying and may make the fingers of a crewmember slip, causing the trigger of an erroneous command. It is possible to cope with such fault by providing interaction techniques that are tolerant to vibrations, e.g. Brace Touch for tactile screens in a cockpit (Palanque et al. 2019).

## Currently Known Solutions and Challenges

In the domain of dependable systems, fault-related issues have been studied and current state of the art identifies four different ways of dealing with them, as presented in (Avizienis et al. 2004):

- *Fault prevention*: avoiding as much as possible the introduction of faults during the development of the system. It is usually performed by following rigorous development processes (e.g. (DO-178C 2012), the use of formal description techniques (Bowen & Stavridou 1993) and (Hamon et al. 2013), proving properties over models, as well as introducing barriers in the designs as proposed in (Hollnagel 2004) and (Basnyat et al. 2007).

- *Fault removal*: reducing the number of faults that can occur. It can be performed i) during system development, usually using verification of properties (Pnueli 1986), theorem proving, model-checking, testing, fault injection, … or ii) during the use of the system via corrective maintenance or fault removal techniques (e.g. starting a redundant system).

- *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 recovery and fault detection are usually achieved by adding redundancy, diversity and segregation thus using multiple versions of the same software in parallel. Fault mitigation is another aspect of fault tolerance, which targets at reducing the impact of faults (when they cannot be removed or prevented).
- *Fault forecasting*: estimating the number, future incidence and likely consequences of faults (usually by statistical evaluation of the occurrence and consequences of faults). This is based on the gathering of real data (both during tests and usage of the system).

One key element with the identification of faults is to make explicit their relationship with failures. Figure 6 describes this relationship. When a fault occurs, it may set a given service of the system in an error state. The system (and its associated service) may come back to a nominal state if the fault is removed. If the fault is not removed and the service is requested, then the failure will occur and it will be noticed that the service is faulty. This failure may, in turn, trigger other faults to other services of the system.

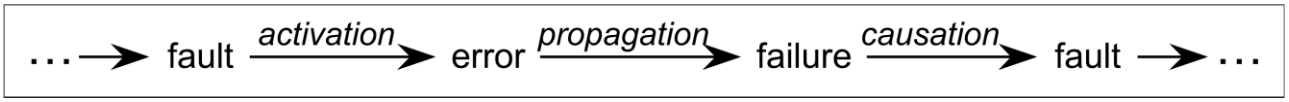


Figure 6. Propagation of faults into error states and possibly failures (from (Avizienis et al. 2004))

In the Psychology and Human-Computer Interaction (HCI) domain, this description of faults, errors and failure is not used to describe human errors. Psychology studies human behavior in order to **understand** the genotype (i.e. the root cause) of human error and to **categorize** them as, for instance, Reason's classification on slips, lapses and mistakes (Reason 1990). HCI focusses on designing and engineering solutions to **prevent and remove** them. Previous paragraphs have been presenting some methods and techniques to address faults induced in the system. We give below some techniques and methods to address faults induced in the human.

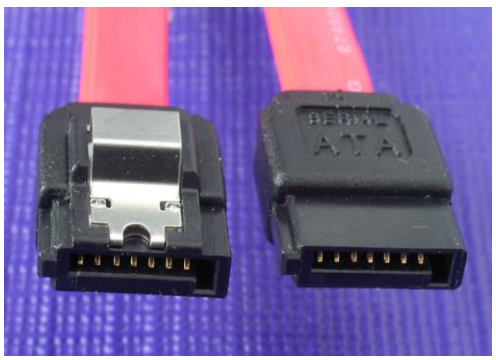

Figure 7. A poka yoke to prevent human-made operational faults induced in the system

Figure 7 presents a hardware barrier to prevent human-made faults to occur when an operator connects a device using the cable and connectors in Figure 7. The red cable is made up of seven inner cables that must be connected in the correct order. The plastic connector aligns the seven cables in such a way that the order cannot be changed (by slip (inattention) or by mistake (incorrect knowledge). However, the entire order could be wrong if the operator uses the connector in an upside down manner. To prevent this, the plastic cover of the connector exhibits a small hole (on its right-

hand side only) that will be filled by a small pic on the connector on the device side. This will prevent such human-made faults and ensure the correct ordering of the inner cable during the connection.

Clearly, this solution is a generic pattern (called Poka Yoke) that should be integrated in a design each time a human-made fault may occur. However, the solution presented requires fine-tuning of this generic pattern to the specific problem.

The fault number 6 in previous section refers to a large category of faults in the human behavior called Cognitive Biases (see (Benson 2016) for structured list of them. According to (Haselton et al. 2005) a cognitive bias can be defined as "Cases in which human cognition reliably produces representations that are systematically distorted compared to some aspect of objective reality". This definition highlights the fact that such behavior is reliable, meaning that human cognition will systematically function (some might say dysfunction) in the same way. The main characteristic of cognitive biases is that they correspond to unconscious behavior of people (usually reacting intuitively to a problem or a situation). The best way to debunk cognitive biases is to make people aware of it. This is the generic solution to the processing of these faults. However, like in the system side faults, to avoid the occurrence of a given cognitive bias requires the design and the implementation of a dedicated solution. In the fault number 6, the solution to debunk the anchoring effect bias is to add in the system a function switching off all the information display. In the field of HCI some specific biases have been studied (e.g. peak-end effect (Cockburn et al. 2019)) and their use for design (e.g. organizing work over multiple pages taking into account peak-end effect) has been proposed. Similarly, work reported in (Saint-Lot et al. 2020), proposes a graphical countermeasure to cognitive tunneling bias (an orange-red flash of 300 milliseconds with a 15% opacity) to improve reaction time of air traffic controller to alarms.

## Conclusion

This paper argues that interactive systems are a specific and complex kind of systems that require dedicated processes, methods and tools for their design, development and validation. With this complexity in mind, we presented a generic hardware-software-operator architecture (called H-MIODMIT) describing the components of interactive systems and their interactions. This paper also presented a generic taxonomy of faults encompassing both faults that affect the operator and faults that affect the interactive system.

We have presented some concrete examples of faults in various components of the interactive systems architecture together with some mechanisms to prevent, remove and tolerate them. While some of these mechanisms are recent (e.g. interaction techniques to remove cognitive biases) other ones are very recent and dedicated to the interaction level of the architecture (e.g. (Cockburn et al. 2017).

Future work will target at addressing the multiplayer perspective. First, the notion of collaborative work, teams and work distribution is not addressed neither in the architecture nor in the taxonomy of faults. Beyond, failures related to combinations of faults are not addressed either. Recent work on fault trees can provide a starting point to this but more work in indeed required.

## References


Albinsson, P.A. & Zhai, S. 2003. High Precision Touch Screen Interaction. Proc. ACM CHI conference, pp. 105-11

Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C. 2004, 'Basic concepts and taxonomy of dependable and secure computing' in IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33.

Basnyat, S., Palanque, P., Schupp, B., & Wright, P. 2007. Formal socio-technical barrier modelling for safety-critical interactive systems design. Safety Science, 45(5), 545-565.

Benson B. 2016. Cognitive biases cheat sheet. https://medium.com/better-humans/cognitive-bias-cheat-sheet-55a472476b18 (retrieved July 2021).

Bowen J. and Stavridou V. Formal Methods, Safety-Critical Systems and Standards. Software Engineering Journal, 8(4):189–209, July 1993.

Card S., Moran T., Newell A. The psychology of human-computer interaction. Erlbaum 1983, ISBN 0898598591, pp. I-XIII, 1-469

Cockburn A., Gutwin C., Palanque P., Deleris Y., Trask C., Coveney A., Yung M, MacLean K. Turbulent Touch: Touchscreen Input for Cockpit Flight Displays. CHI 2017: 6742-6753

Cockburn A., Quinn P., and Gutwin C. 2015. Examining the Peak-End Effects of Subjective Experience. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15). ACM DL, 357–366.

Cronel M., Dumas B., Palanque P., Canny A. 2019. MIODMIT: A Generic Architecture for Dynamic Multimodal Interactive Systems. In Human-Centered Software Engineering. HCSE 2018. LNCS, vol 11262. Springer

Diaper D. & Stanton N. The handbook of task analysis for human-computer interaction. Lawrence Erlbaum Associates, 2003. ISBN 0-8058-4432-5

DO-178C / ED-12C, Software Considerations in Airborne Systems and Equipment Certification, published by RTCA and EUROCAE, 2012

Fitts P. The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology. 1954; 47:pp. 381-391.

Hamon A., Palanque P., Silva J., Deleris Y., & Barboni E. 2013. Formal description of multi-touch interactions. Symp. on Engineering int. computing sys. (EICS '13). ACM DL, 207–216.

Haselton MG, Nettle D, Andrews PW .2005). The evolution of cognitive bias. The Handbook of Evolutionary Psychology. Hoboken, NJ, US: John Wiley & Sons Inc. pp. 724–746

Hollnagel, E. Barriers and Accident Prevention. 2004. Ashgate.

Murre JMJ, Dros J. 2015. Replication and Analysis of Ebbinghaus' Forgetting Curve. PLoS ONE 10 (7). https://doi.org/10.1371/journal.pone.0120644

Navarre D., Palanque P., Bastide R., Sy O. 2001. Structuring Interactive Systems Specifications for Executability and Prototypability. Interactive Systems Design, Specification, and Verification. DSV-IS 2000. LNCS, vol 1946. Springer.

Olwal, A. & Feiner, S. Rubbing the Fisheye: Precise Touch-Screen Interaction with Ges-tures and Fisheye Views. Conference Supplement of UIST 2003. pp. 83-84.

Palanque, P., Cockburn, A., Désert-Legendre, L., Gutwin, C., Deleris, Y. 'Brace Touch: A Dependable, Turbulence-Tolerant, Multi-touch Interaction Technique for Interactive Cockpits'. Proc. of SAFECOMP 2019, LNCS, Springer, 53-68.

Palanque, P., Cockburn, A., Gutwin, C. 2020, A Classification of Faults Covering the Human-Computer Interaction Loop. SAFECOMP, LNCS n° 12234, pp.434-448, 2020, Springer.

Pnueli A Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends. LNCS n° 224 p.510-584. Springer Verlag 1986.

Reason J. Managing the Risks of Organizational Accidents. Ashgate Publishing limited. 1997.

Saint-Lot J., Imbert J-P. and Dehais F. 2020. Red Alert: A Cognitive Countermeasure to Mitigate Attentional Tunneling. ACM CHI '20 Conf. ACM DL.

Soukoreff W. & MacKenzie S. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. IJHCS. 61(6): 751-789 (2004)

# Biography

**Author Name**. Célia Martinie is associate professor in Computer Science at the Université Paul Sabatier Toulouse III. She obtained her PhD in 2011, contributing to the engineering of safe and usable interactive systems. She has a long-term experience in the area of interactive critical systems, including design and implementation issues of command and control systems. Her main research skills and knowledge target methods, processes, techniques and tools to engineer critical interactive systems and for their integration within an organizational context for safe operations. She has been working closely with main actors in the area of safety critical interactive systems including Airbus for aircraft cockpits, CNES and ESA for satellite ground segments, and EUROCONTROL for air traffic management. She has published more than 60 articles in peer-reviewed international conferences and in journals.

**Philippe Palanque**. Dr. Philippe Palanque is Professor in Computer Science at the University Toulouse 3 “Paul Sabatier” and is head of the Reliability of Systems and Software department at the Toulouse Research Institute on Computer Science (IRIT) in France. Since the late 80s, he works on the development and application of formal description techniques for interactive system. The main driver of Philippe’s research has been to address in an even way Usability, Safety and Dependability in order to build trustable safety critical interactive systems. He is the chair of the IFIP Technical Committee 13 on Human-Computer Interaction and the former adjunct chair for specialized conferences at ACM SIGCHI. As for 2022 he is the co-paper chair of ACM SIGCHI Engineering Interactive Computing Systems conference.