Prédiction des conflits dans des systèmes homme-machine

S. Pizziol

▶ **To cite this version:**

S. Pizziol. Prédiction des conflits dans des systèmes homme-machine. Interface homme-machine [cs.HC]. Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), 2013. Français. NNT : . tel-01068870

HAL Id: tel-01068870

https://theses.hal.science/tel-01068870

Submitted on 26 Sep 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)

Présentée et soutenue par :
Sergio PIZZIOL
le vendredi 29 novembre 2013

Titre :

Prédiction des conflits dans des systèmes homme-machine

École doctorale et discipline ou spécialité :
EDSYS : Systèmes embarqués et Automatique

Unité de recherche :
Équipe d'accueil ISAE-ONERA CSDV

Directeur(s) de Thèse :

Mme Catherine TESSIER (directrice de thèse)
M. Frédéric DEHAIS (co-directeur de thèse)

Jury :

M. Frédéric VANDERHAEGEN - Président
M. Gilles COPPIN - Rapporteur
M. Frédric DEHAIS - co-directeur de thèse
M. Didier DUBOIS
M. Axel SCHULTE - Rapporteur
Mme Catherine TESSIER - Directrice de thèse

Conflict prediction in human-machine systems

Sergio Pizziol

Contents

Introduction

On 31 May 2009, the Airbus A330 flight AF 447 took off from Rio de Janeiro Galeao airport bound for Paris Charles de Gaulle. (...) At around 2 h 02, the Captain left the cockpit. At around 2 h 08, the crew made a course change of 12 degrees to the left, probably to avoid returns detected by the weather radar.
At 2 h 10 min 05, likely following the obstruction of the Pitot probes by ice crystals, the speed indications were incorrect and some automatic systems disconnected. The aeroplane's flight path was not controlled by the two copilots. They were rejoined 1 minute 30 later by the Captain, while the aeroplane was in a stall situation that lasted until the impact with the sea at 2 h 14 min 28.
The accident resulted from the following succession of events:

- *temporary inconsistency between the measured airspeeds, likely following the obstruction of the Pitot probes by ice crystals that led in particular to autopilot disconnection and a reconfiguration to alternate law*
- *inappropriate control inputs that destabilized the flight path*
- *the crew not making the connection between the loss of indicated airspeeds and the appropriate procedure*
- *the PNF's late identification of the deviation in the flight path and insufficient correction by the PF*
- *the crew not identifying the approach to stall, the lack of an immediate reaction on its part and exit from the flight envelope*
- *the crew's failure to diagnose the stall situation and, consequently, the lack of any actions that would have made recovery possible*

The BEA has addressed 41 Safety Recommendations to the DGAC, EASA, the FAA, ICAO and to the Brazilian and Senegalese authorities related to flight recorders, certification, training and recurrent training of pilots, relief of the Captain, SAR and ATC, flight simulators, cockpit ergonomics, operational feedback

and oversight of operators by the national oversight authority.

This is an extract from the synopsis of the BEA (Bureau d'Enquêtes et d'Analyses) Final Report on the flight AF 447 Rio de Janeiro - Paris accident. In the conclusion of the same document it is stated among the findings that:

- *in the absence of a display of the limit speeds on the speed tape on the PFD, the aural stall warning is not confirmed by any specific visual display*
- *the stall warning sounded continuously for 54 seconds*
- *neither of the pilots made any reference to the stall warning or to buffet*
- *the aeroplane's angle of attack is not directly displayed to the pilots*

And among the causes of the accident, the following ones are mentioned:

- *the crew not identifying the approach to stall, their lack of immediate response and the exit from the flight envelope*
- *the crew's failure to diagnose the stall situation and consequently a lack of inputs that would have made it possible to recover from it*
- *task-sharing that was weakened by incomprehension of the situation when the autopilot disconnection occurred (...)*
- *the lack of a clear display in the cockpit of the airspeed inconsistencies identified by the computers*

Therefore among the causes of the accident the BEA identified the crew's wrong situation assessment, in part due to the inconsistencies in the shown information, to the warnings that were not perceived and to the lack of relevant information in the displays.

In 2002 a Massachusetts Institute of Technology study [VJ02] found 184 incidents attributed to situation awareness [Wic08] problems in NASA Aviation Safety Reporting Systems.
In aviation psychology, the analysis of air safety reports [Hol03] has shown that the occurrence of a cognitive conflict is a remarkable precursor to the degradation of human operators' performance, provoking plan continuation error [Ora01].

Experimentations conducted in flight simulators have revealed that such conflicts could lead to patterns of behaviours that indicate perseveration [DTC03, DTCR09a]. This particular behaviour is defined – within the psychology literature – as the tendency to continue or repeat an activity after the cessation of the original stimulation, and to an extent that the activity is often no longer relevant to the task at hand. More precisely, Sandson and Albert [SA84] identified three distinct categories of perseveration, among which the stuck-in-set perseveration, "the inappropriate maintenance of a current category or framework". Once caught up in a perseveration behaviour, it is assumed that most of the human operators' resources are summoned up toward conflict solving. As a consequence, the cognitive abilities of the operators are impaired with a strong tendency for attentional tunneling that can lead to excessive focusing on one display, to the neglect of other information (e.g., alarms) that could question their reasoning. Conflicts not only occur between humans, but may also be induced while interacting with artificial systems. Indeed, similar attentional issues have been widely described within crew-automation conflicts known as 'automation surprises' [SW95, SWB97] whereby the autopilot does not behave as expected by the crew. This cooperation breakdown can lead to accidents with an airworthy airplane where the crew persists in solving a minor conflict [Bil96] "instead of switching to another means or a more direct means to accomplish their flight path management goals" [WS00], and this can occur despite the onset of auditory alarms [BHJ99]. Such hazardous situations are not only relevant in aviation but also in the context of human supervisory control of unmanned vehicles (UVs) where careless design of authority sharing [Ina03] degrades the human operator's performance leading to inadequate behaviours [PW08, VGdVKT06]. Moreover, some authors [Mey01, PW08, Ric09] revealed that unreliable diagnosis automation (i.e. miss-prone vs. false alarm-prone automation) and automation complacency might lead to conflictual situations that also negatively impact attentional resources [MP05, WDGH05] and deteriorate the human operator's global performance [DWM07, WD07]. Approaches such as adaptive automation [She11] and cognitive counter-measures [DCT11] attempt to solve the problem of attention allocation; however, challenges in their implementation still remain. In particular, a critical aspect of adaptive aiding system is to provide help in a timely and accurate matter [dVP11]).

Therefore developing methods and systems that can mitigate those issues is essential. The work that has been carried out during this PhD is part of the research dedicated to human-machine misunderstandings, automation surprises [Pal95] and conflicts[1] between the human and the machine that may arise from such situations.

[1] For the conflict definition see section 2.3.

More precisely the work focuses on models and tools for the prevention, detection, identification and solving of human-machine bad interactions, especially in the aeronautics field[2].

We will propose a formal method dedicated to conflict prediction in human-machine systems to *a priori* identify internal state changes that are likely to bring about conflicting situations. The achievement of this main objective will then lead us to to propose hints for the further development of a real time conflict detection model.

The document is composed of six chapters.

In the **first chapter** we define the context of the work and detail the architecture of the system we deal with.

In the **second chapter** a survey of conflict unified definitions in the Artificial Intelligence and Cognitive Science literature is carried out. Keeping in mind the key concepts identified in the literature a conflict taxonomy and definition is proposed. In the last section of the chapter we review the existing techniques and concepts that are developed to prevent bad human-machine interactions.

In the **third chapter** we propose a formal method based on Petri net modelling to identify human-machine interaction designs that are likely to create conflicts. The proposed model comes from the analysis of two real cases. This method is a tool to help designers. More precisely it is an *a priori* conflict *prevention and solving* tool: once the vulnerabilities highlighted, the interaction design may be corrected.

In the **fourth chapter** we detail an experiment that has been conducted in a flight simulator in order to show the relevance of the method proposed in the third chapter.

Even if the formal Petri net method does not highlight any weakness, conflicts may still arise in some cases. In the **fifth chapter** we propose an enhanced model with uncertainty management to be used as a real time conflict *detection* and *identification* tool.

[2]The system we will detail the more is the pilot/auto-flight system (AFS) consisting of the pilot, the autopilot (AP), the auto-thrust (ATHR), the flight envelope system (ENV) and the Flight Management System (FMS) controlling a modern commercial plane. Another system that will be studied is the human–machine system controlling an unmanned vehicle.

This uncertainty model could possibly be further enriched thanks to data coming from the observation of the human operator, and methods to infer the human operator's "state". As an example a fuzzy model to characterize attentional tunnelling is presented in the **sixth chapter** .

We have identified the lack of reversibility (of human actions on the machine state) as a source of possible human-automation bad interactions. In the **seventh chapter** we define a reversibility scale and we develop a reversibility check in order to assess the degree of reversibility of the human operator's actions.

In the latter part of this document the **conclusions** are drawn and a synthesis of the work is provided. The prospects for further work are provided as well.

Chapter 1

Context

1.1 Introduction

In this chapter we define the context of the work. In the first section we detail the architecture of the system we deal with. In the second section we define the authority sharing relation between two agents. In the last section we focus on the human-machine system with a conflict manager.

1.2 The human-machine system

A *human–machine system* (see figure 1.1) is a two-agent team formed by a *human operator* and a *machine* with a common *goal* [Jen95], they *communicate* and act on a *physical system* (or just *system*) for the achievement of their goal. The goal achievement is pursued through the execution of *functions* [MC99]. Some of those functions can only be executed by the human operator, some only by the machine. In this work we focus on human–machine systems controlling a vehicle, the *human* being either an on-board *pilot* or a *remote controlling operator*[1]. Nevertheless the concepts developed in this chapter are applicable in the wider context of human-machine systems.

We will call *machine* the set of all the automation systems that can perform at least one function. The designer of the human–machine system, when defining the functions, also defines the perimeter of the machine.
The human/machine interface (H/M interface) is composed by input devices (e.g. buttons, knobs etc.) and output devices (e.g. displays, visual and aural alarms etc.).

[1] We will not consider several humans (crew) controlling the system.

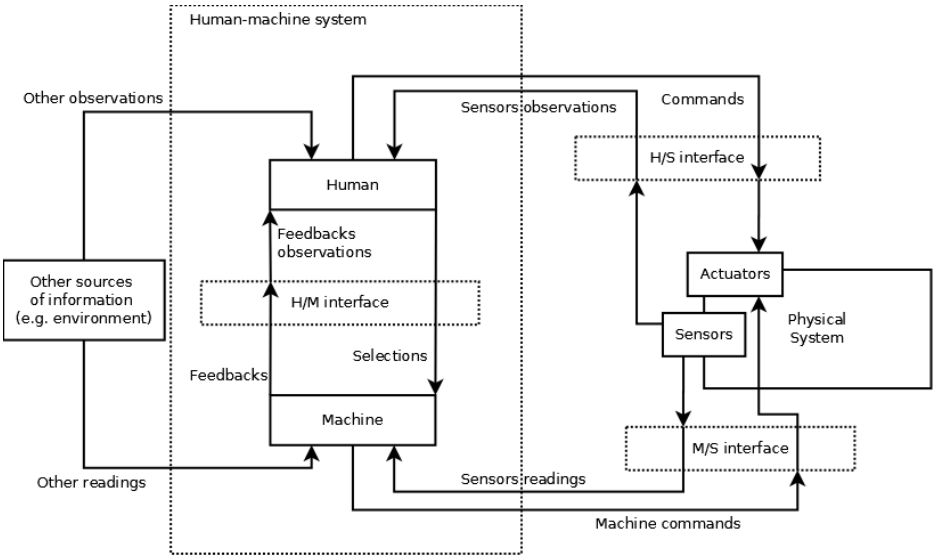


Figure 1.1: Human–machine systems in charge of controlling a physical system

We call *feedbacks* the machine to human communication through the human/machine interface.
The *selections* are human actions that constitute the human to machine communication through the human/machine interface.
We call *internal state* the state of the machine. In principle feedbacks allow a certain degree of observability on the internal state so that the human could make *observations*; selections should affect the internal state. Indeed we call *selection triggered internal state changes*, or *selection changes*, internal state changes that are the consequence of selections. We call *automated internal state changes*, or shortly *automated changes*, internal state changes that are not caused by selections.
The human/physical system interface (H/S interface) is composed of input devices (e.g. control stick, yoke, control pedals, etc.) and output devices (e.g. sensors displays, control stick with retrofeedback etc.).
The machine/physical system interface (M/S interface) is composed of the software and the hardware systems meant to provide the connection between the machine and the sensors and actuators of the physical system.
We define the *sensors* as a part of the controlled physical system providing information about the physical system state itself through the H/S interface and the through the M/S interface. For instance we consider a Global Positioning System

(GPS) as a sensor. We do not consider aerials dedicated to communication[2] as sensors.
We define the *actuators* as parts of the controlled physical system that can change the system state and are controlled through the H/S interface and the M/S interface. We consider the servoing systems (as the automatic control loops) as part of the actuators.
Commands are human actions on the H/S interface meant to control the actuators. Similarly *machine commands* are data sent from the machine to the actuators through the M/S interface.
Sometimes, for ergonomics reasons, a single physical action may embed *commands* and *selections*: that is the case for an operator taking the authority on the steering wheel of a unmanned ground vehicle (a *selection*) and controlling the direction of the vehicle (a *command*) with a single movement on the control stick. Sometimes the same display may be part of the H/M interface and the H/S interface at the same time.
We call *other observations* all other pieces of information the human receives neither from sensor observations nor from feedback observations (e.g. a visual estimate of the altitude based on the observation of the ground, communications received by the radio coming from the air traffic control (ATC)). Similarly we call *other readings* all other pieces of information the machine receives neither from sensor reading nor from selections (e.g. radio communication form the traffic collision avoidance system (TCAS) of another aircraft).
The underlying architecture is a top-down functional decomposition [MC99] (see figure 1.2). A function may be performed by the human or by the machine (respectively *human-function* and *machine-function*). In the functional decomposition we use the operators AND, OR.

There are interdependencies between functions (as in the structured analysis and design technique, SADT [MM87, Myl04]): a function may receive *ancillary inputs*, may provide *outputs* and may be *controlled* by other functions (see figure 1.3).
Ancillary inputs are pieces of information that are meant to help during the execution of the function but are not necessary needed.
Controls are pieces of information that usually represent the reference for the functions. *Controls* are needed for the function execution.
Outputs are pieces of information generated by the function execution and that are used as *ancillary inputs* or *controls* by other functions.
An agent executing a function may need the *allocation of some non co-usable resources*. A *non co-usable resource* is a physical object that can be allocated to only

[2]Not used for radiolocation.

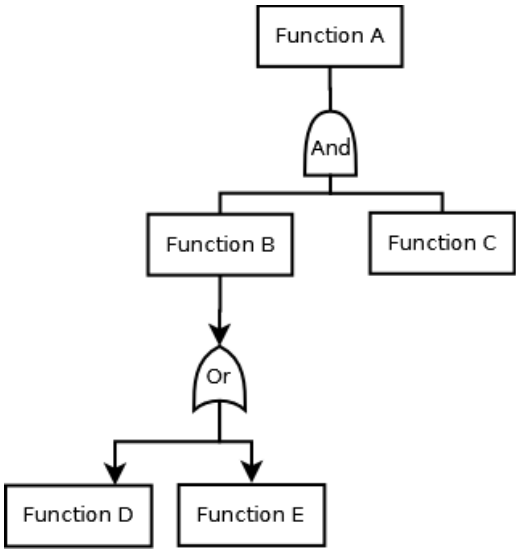


Figure 1.2: Function logical decomposition

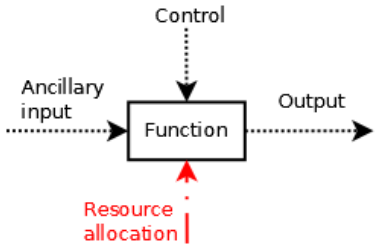


Figure 1.3: Functions in SADT (structured analysis and design technique).

one agent at the time (e.g. an actuator)[3].
For a wise function allocation the human-machine system designer should keep in mind the human and machine strengths and weaknesses. The machine is good at performing repetitive tasks and at quick and precise computation, but is bad at unexpected event management ([LSMC10] in [Mer11]). The human is usually better in *adaptability*, i.e. in failure and unexpected event management and in coping with uncertainty and lack of information ([Ras83] in [Mer11]).
Remember that controls are references to follow and that they are strictly needed

[3]Sometimes to be short we refer to non co-usable resources as just *resources*, in this work we do not deal with resources that may be used simultaneously by several agents. Hereafter we will refer to the resource sequential use by several agents as *resource sharing*: that has nothing to do with the resource simultaneous use by several agents.

for the function execution. Indeed data coming from a machine-function are usually ancillary inputs for a human-function (e.g. the flight director indications) because:

defining an input as a control for a human-function contradicts the fact that the human may usually cope with lack of information

the human is not as good as the machine in precisely following a reference (in nominal conditions if precision is required it is wise to define the function as a machine-function)

On the other hand in nominal conditions instructions coming from a human-function are usually controls for a machine-function, the machine being good at precise instruction execution and bad at dealing with lack of information.

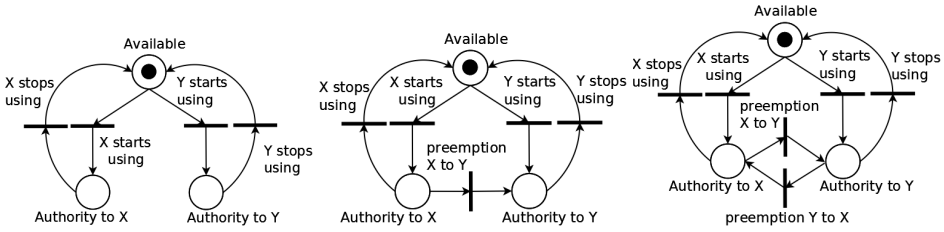
1.3 Authority sharing

Authority sharing is a relationship that must necessarily be defined when a non co-usable resource could potentially be requested by several agents. This relationship allows to answer the question: "what happens if a resource is allocated to an agent and subsequently asked for by another agent?". [MTD10a] details a two-agent authority sharing relationship (see table 1.1). Each agent may have no access to the resource at all, simple access or access with pre-emption rights. Four cases are described by [MTD10a]: the degenerate case for which just one of the two agents has access to the resource, the case in which both agents have simple access with no pre-emption (*cooperative sharing*), the case in which just one agent has pre-emption rights (*exclusionary sharing*) and the case in which both of them have pre-emption rights (*preemptive sharing*). Note that an agent may be interrupted if and only if the other agent has pre-emption rights. A Petri net representation for those relations is given in figure 1.4, where the *available* place is marked if no agent is using the resource. An evaluation of the characteristics of each case is necessary in order to choose the solution that best fits the intended use of the resource.
In *cooperative sharing* each agent waits patiently for the resource to be available to take over (see figure 1.4a). In this case there are no interruption issues. It is worth noticing that this solution is not adapted when the first agent has to perform short and compelling tasks and the second one performs not compelling and long tasks: the first agent could fail to perform their task waiting for the second to release the resource. In this case the *exclusionary sharing* is a well suited solution,

with pre-emption right to the agent performing the compelling task. In this case the problem of the interruptions suffered by the second agent has to be evaluated (see figure 1.4b, transition preemption agent X to agent Y). In cooperative and exclusionary sharing the agents should provide a mechanism for the resource release in order to avoid mutual deadlocks.
The *preemptive sharing* is a particular case: if the agents do not follow rules to limit their mutual interruptions (see figure 1.4c) a dangerous situation called *authority oscillation* may occur [MTD10a], on the other hand there is no deadlock risk.
Note that the *authority relation* may evolve in time, passing from a sharing relation to another. For instance that is the case if an authority manager is in charge of preventing and solving the arising problems.

Agent	Authority	Access	Preemption	Interruptions	Description	Symmetric relation
X Y	No access Preemption	No Yes	- Yes	- No	Not Sharing	No
X Y	Access Preemption	Yes Yes	No Yes	Yes No	Exclusionary Sharing	No
X Y	Access Access	Yes Yes	No No	No No	Cooperative Sharing	Yes
X Y	Preemption Preemption	Yes Yes	Yes Yes	Yes Yes	Preemptive Sharing	Yes

Table 1.1: Two-agent authority relations to share a resource



(a) Cooperative sharing. (b) Exclusionary sharing. (c) Preemptive sharing.

Figure 1.4: A Petri Net representation of the authority relations.

1.4 The human-machine system with a conflict manager

In this work we will consider the human-machine system as a two-agent system (see figure 1.1), where both agents can perform actions so as to control the physical system, which may be subject to uncontrolled events (e.g. failures). Note that the machine is considered an agent because some internal state changes and automated commands can be performed by the machine itself without prior consent of the human, and sometimes despite the human's actions.
Conflicts in a human-machine system stem from the fact that both agents can decide and act on the physical system and their actions may not be consistent, either because the expected plan for the human or the machine is not followed anymore, or because the human has a wrong situation awareness [Wic08], or both. In order to prevent a mission degradation, the agents' plans, and possibly the authority allocation (i.e. which agent is controlling which resource), have to be adapted [MTD10b]. This is a real challenge as in human-machine systems the human agent is hardly controllable and no "model" of the human's decision processes is available.

We will propose a formal method dedicated to conflict prediction in human-machine systems to *a priori* identify internal state changes that are likely to bring about conflicting situations. Then we will propose hints for the further development of a real time conflict detection model to be included in a *conflict manager*. A possible general architecture of a human-machine system equipped with a *conflict manager* is shown in figure 1.5. The red part of the diagram represents a conflict manager that is only based on the actions performed by the human operator and on the feedbacks sent by the automation. This could be enhanced thanks to additional information coming from special sensors dedicated to the observation of the human operator. In this case the architecture would include the green part of the diagram. Note that *cognitives countermeasures* (or *countermeasures*) [DCT11] are pieces of information sent through the human/machine interface that are meant to help the human to better understand and solve the conflicts. *Reconfigurations* are pieces of information sent to the machine to change its behaviour in order to help conflict solving[4].

[4]For instance a reconfiguration could be to switch the machine to a degraded mode.

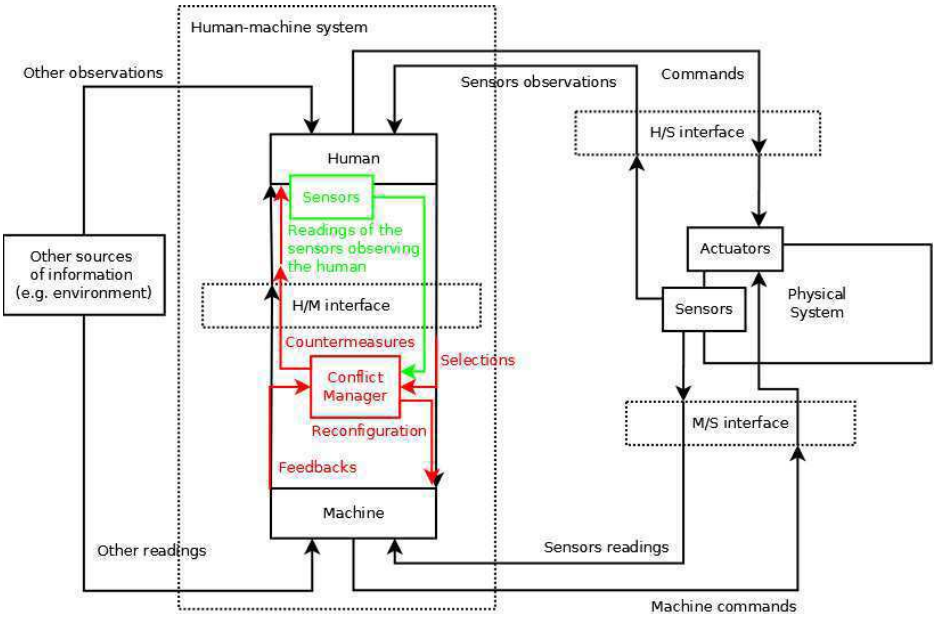


Figure 1.5: Human–machine systems with conflict manager

Chapter 2

Conflicts

2.1 Introduction

In section 2.2 a survey of conflict definitions in the Artificial Intelligence and Cognitive Science literature is carried out. This survey is not meant to be a complete review of the definitions of conflicts, but it gives examples of some *unified* definitions of conflicts. As we studied these examples we had to come up with the idea that those *unified* definitions do not help us solving conflicts.
Different unified definitions focus on different aspects of conflict depending on both the nature of the agents involved (human agents of artificial agents) and the epistemological point of view (empirical, conceptual, formal and methodological) [TMFC00].
In section 2.3 we give our conceptual conflict definition that openly waives any unifying and generalizing ambition.
In section 2.4 we present some approaches for conflict prevention in the domain of human-machine interaction. More details about some of those works are given in order to take cues for our further developments.

2.2 Unified conflict definitions

In Distributed Artificial Intelligence conflict is often related to the concept of logic inconsistency [MD00]. Conflict occurrence is considered as an obstacle to get a solution. For instance in cooperative multi-agent systems conflict is seen as a *non cooperative* situation. This sort of definition *by negation* is common to many authors.

The first conflict definition we analyse is given in [Cas00, CF00]:

Conflict: *is a situation in which three conditions are true:*

the agents have at least two contradictory goals

the agents are aware of their goals to be contradictory

the agents have to make a choice

So according to the authors all conflicts are incompatibilities between the agents' goals. Further on the authors observe that conflicts may arise for other reasons than the agents having two contradictory goals: *differences in the knowledge of the agents or because of a resource*.
To make their definition fit with conflicts due to differences of knowledge the authors add as a goal the fact that:

G1: *all the agents should have coherent beliefs*

In some cases imposing this goal to the agents is over-constricting as the coherence for *relevant* information should be enough [DP00]. Moreover this requirement is unattainable if the agents do not have the same state representation for the same state variable.

To make their definition fit with conflicts due to resource sharing the authors state that *"one can derive that the first agent will have one additional goal: the negation of the second agent's goal in competition for the resource"* [CF00], i.e. they define as a goal:

G2: *the negation of the second agent's goal in competition for the resource*

The same concept is expressed by the latin motto "Mors tua, vita mea", that means "You must die so that I may live". This expression is used when attempting to reach a goal where there can be only one winner: the other competitor's failure constitutes a prerequisite for success. One may object that one agent, in pursuing as a goal the negation of another agent's goal, is actually being committed to a side effect of its own goal achievement. For instance in the sentence *taking an aspirin to decrease the body temperature, that will make me sweat* the agent is committed in body temperature decrease (they *intent it* [CL90]), and they are aware but not committed in sweating (they *do it intentionally* [CL90]).

In conclusion all the conflicts may be boiled down to logical inconsistency between goals thanks to auxiliary goals. That unified definition does not help solving conflicts. As a consequence of **G2** the agents needing of the same resource do not even try to negotiate about a delayed or partial goal achievement, maliciously taking the other's in-satisfaction as a metric of their own satisfaction. The solving strategy consisting in giving the agents more available resources, which the authors contemplate for the resolution of social conflicts, would be ineffective if the agents make no difference between their own goals and the side effects.

For [Han00] the conflict is defined as:

> **Conflict:** *a situation in which the requirements of an agent are not compatible with the requirements of other agents.*

For the authors conflicts *"arise if there are goals to achieve that conflict in some way, for instance because of competing for scarce resource"*. Consequently *requirements* should model both *goals that are potentially in conflict* and the *need for resources*. This conflict definition unifies conflicts thanks to the *requirements* definition.

[DP00] introduce the concept of propositional attitudes (PA) in order to generalize the definition given by [CF00]. PAs are propositions that express the agents' goals, the need for resources, the nature of the resources, rules about the logic of the system and physical constraints to be respected. PAs specifying relations between other PAs are called *crucial PAs*. An agent may hold some PAs and crucial PAs. Holding a *goal PA* means trying to achieve the goal expressed by the corresponding PA, holding other kinds of PAs means believing the propositions expressed by those PAs. A *context* is the union of all the PAs held by the agents. For [DP00] a *conflict* is defined as:

> **Conflict:** *a context in which at least one crucial PA, evaluated using the values of the other non crucial PAs of the context, is falsified.*

Crucial PA are the keystone of this definition: they express what *matters* case by case. Nevertheless the given definition is a unified definition of conflicts. Keeping this conflict definition in mind we can say that crucial PAs express meta-rules defining relations about the goals and believes (that *matter*).
Typical crucial PAs are "the believes and the goals of the agents must be logically consistent" or "the resource R is not shareable". It is possible to express any kind of crucial PA. For instance it is possible, but doubtfully useful, to define a crucial

PA as "the believes and the goals of the agents must be logically inconsistent".
Therefore all the conflict cases (e.g. conflicts for non shareable or depletable resources, conflicts between goals, conflicts between beliefs) need to be described by appropriate crucial PAs (representing the informative part of the conflict definition). The strength (and the limit) of this definition is its generality: every conflicting situation could (and should) be modelled (from scratch).
In conclusion crucial PAs may express conflicts due to contradictory goals that matter, differences in knowledge that matter and resource whose shortage matter.

[Mer11] uses the concept of *resource* as a key of the definition of the conflict. A resource can be a physical resource, an information, a goal or a constraint. Agents make *plans*. Each *plan* is the explicit definition of all the resources needed by the agents to try to achieve their goals. A *plan* is more than a simple list of *resources*, it shows the interdependencies between the *resources*. A resource may be allocated by an agent to another *resource*. *Resources* to which other *resources* are allocated and that are not allocated themselves are defined as *goals*. A change in the *plan* may lead to a situation for which a *resource* changes allocations.

[Mer11] defines a conflict as:

Conflict: *a change in the plan that leads to the loss of a goal.*

For the author conflicts arise because a non shareable *resource* has been preempted by another agent for a new allocation or because the *resource* has been destroyed. In this model no formal representation of the goal semantics is given: consequently the occurrence of goals *logical inconsistencies* can be detected only if some *ad hoc constraint resources* are defined by the designer. If so the arising goal logical inconsistencies result from the destruction of the constraint resource defined for this purpose. The way agents build their plans is not part of the model. For that reason even in the case where some constraint resources were defined, no explicit logical consistency forecast is possible before the problem arises, i.e. before the *ad hoc* constraint resources are destroyed. No formal representation of the agents' beliefs coherence is given either. In conclusion this definition is well suited for conflicts due to critical *resource* management, but is not well suited to check goal logical inconsistencies and belief inconsistencies.

In all those definitions the authors focus on a unified conflict definition, sometimes expressing the exclusive access to resources as a goal [CF00], sometimes defining the need for belief coherence as a goal [CF00], sometimes expressing goals as resources [Mer11], sometimes expressing beliefs, goals and need for re-

sources as a new and general concept [DP00].

[Deh12] gives a definition that goes towards a taxonomy of conflicts cases respecting their different natures:

> **Conflict:** *a situation in which either*
>
> - *a non co-usable resource is the reason for a competition to arise*
> - *at least two different pieces of* relevant *information are contradictory*

In the next section we will give a conceptual conflict definition and a conflict taxonomy that goes in the same way as [Deh12]. This definition **openly waives any unifying and generalizing ambition** for the benefit of a greater adaptation of the proposed solution to the problem[1].

2.3 Conflict conceptual definition

We define an agent's action as an *effective action* if, considering the agent's actual believes and goals, it is coherent with a predefined logic accepted by the multi-agent system designer[2]. In other words an agent's action is effective if it is locally (at the agent level) coherent with the agent's accepted behaviour. The given effectiveness definition is less constricting than others that are commonly used in literature, for instance [CL90] in [Jen95]: *if an agent intended actions are executed in a world in which its beliefs are true, the desired state of affairs should ensue*. In our case we do not require the "theoretical goal achievement" but just the "coherence with a predefined norm" in order to have a definition that is sound also in unexpected situations. A good example is the case of a pilot/flight management system in control of a modern civil aircraft with loss of the thrust in both engines [Boa10]. In a situation like this it is hard to tell if an agent's intended actions, even if their beliefs are true, will bring to the desired state of affairs. The respect of a predefined logic for an autopilot means that the software and hardware are not

[1] We decided to take the advice of Wittgenstein [BM11] *(he) repeats: "Don't think but look!" [Wit53]; and such looking is done vis a vis particular cases, not thoughtful generalizations. In giving the meaning of a word, any explanatory generalization should be replaced by a description of use. The traditional idea that a proposition houses a content (...) gives way to an emphasis on the diversity of uses.*

[2] If one agent's actions are *non effective* (e.g. because of a failure) either a local diagnosis and reconfiguration are possible, or not (e.g. human operator's procedural error). In the latter case the non effective actions are likely to be also incoherent actions. In this study we are not going to deal with this case, i.e. the agents are respecting a predefined and accepted logic.

faulty, for a pilot that means that he is following the procedure.

In the frame of a multi-agent system we define a *conflict* as:

Conflict: *the execution of actions that are* effective *but in spite of this are either* logically incoherent*, either* physically incoherent *or* epistemically incoherent.

The requirement on the actions effectiveness is meant to exclude mere agents' errors from the conflict definition. Our definition embeds three terms defined as follows:

•Logically incoherent [SM06]: at least two goals are logically contradictory, the agent performing the actions have opposite desires. *Example: two agents are in charge of the vertical control of an aircraft. The altitude is 4000 ft. One wants to climb to 6000 ft and the other one wants to descend to 2000 ft.*

•Physically incoherent [TMFC00, SM06]: at least a depletable or not shareable resource (e.g. a physical object) is the cause of a competition, the agents preemptively take over the resource. *Example: one agent is in charge of the vertical control of an aircraft and another agent is in charge of the longitudinal control. Taking over the authority of the same flight control surfaces (e.g. the spoilerons*[3]*, that could affect the roll and the climbing rate) at the same time is a physically incoherent action.*

•Epistemically incoherent [TMFC00]: the agents performing the actions do not share the same point of view on at least two *relevant* pieces of information. *Example: two agents are both in charge of the vertical control of an aircraft. They both want to reach altitude 5000 ft. One agent estimates the current altitude to be 6000 ft and the other one 4000 ft.*

2.3.1 Conflict taxonomy

We propose thereafter a conflict *taxonomy* that relies on the three preceding inconsistencies.

Logical conflicts are when the agents' goals are logically contradictory. Note that contradictory goals are not necessarily incompatible: despite the fact that the

[3]Spoilerons are flight control surfaces, specifically spoilers that can be used asymmetrically to achieve the effect of ailerons.

agents' incapability to evaluate a trade-off could hinder the solution of the conflict, an acceptable compromise solution may still exist. Negotiation techniques have been proposed to solve this kind of conflict [Kra97].

Physical conflicts are when the agents' goals are incompatible because of the resources required to achieve their goals. In this case a wise resource sharing is needed.

Knowledge conflicts are when the agents' goals are coherent [Wil83, SM06], but the agents' information for decision-making about how to achieve their goals is not the same. Such conflicts may concern the agents' situation assessment and procedures. A particular case of knowledge conflict is the wrong human assessment of the internal state of the machine.

Definition	**Suited to model physical conflicts**	**Suited to model logical conflicts**	**Suited to model knowledge conflicts**	**Remarks**
[Cas00, CF00]	No, the definition is such that the conflict cannot be solved	Yes	Yes, but too constrained	All the conflict are considered as logical
[Han00]	Yes	Yes	No	Not adapted to model human-machine systems
[DP00]	Yes	Yes	Yes	Difficult implementation, too general
[Mer11]	Yes	Yes, but prevention is not possible	No	All the conflict are considered as physical

Table 2.1: Comparison between conflict definitions.

Physical conflicts arise because of resource sharing authority issues (see section 1.3). Logical and knowledge conflicts are likely to bring about resource sharing authority issues too.

Relations between conflict unified definitions and conflict taxonomy

We now review some of the assumptions used for the conflict definitions given in section 2.2 with regard to the conflict taxonomy we propose (see table 2.1).
We can say that [Cas00, CF00] transform knowledge conflicts in logical conflicts imposing goal **G1** that imposes the total matching between the believes of the agents.1 Consequently this definition is too constrained to model knowledge conflicts.

[CF00] transforms also physical conflicts in logical conflicts imposing goal **G2** . That assumption makes this definition unsuited to model physical conflicts.
[Han00] model physical conflicts and logical conflicts *via* resource constraints and goal constraints, however no representation of the differences between the agents' knowledge is given, so the definition is unsuited to model knowledge conflicts.
[DP00] model physical, knowledge and logical conflicts *via* specific crucial PAs. The definition is adapted to model all kinds of conflicts but it is far too general to be handy.

The objective of this work is the prevention and detection of conflicts due to cognitive issues (see Introduction). More precisely we focus on the human assessment of the internal state of the machine, and on *knowledge conflicts regarding the internal state*: for the sake of simplicity starting from now we will refer to them as just *conflicts*[4]. Note that the study of the other kinds of conflicts is out of the scope of this work.

2.4 Human-machine conflicts prevention

Knowledge conflicts regarding the internal state occur when the human believes that the *internal state* has a value that is different from the actual one, and consequently makes inappropriate requests or responses to the machine [JMH03].
In the frame of conflict prevention in the human-machine interaction literature many approaches have been proposed. Some of them are formal approaches focused on the observability of the internal state.

> **Observable finite state machine (FSM) [OW90]:** *The finite state machine (FSM) representing the machine and the H/M interface is* observable *if the human, thanks to the feedbacks, can have a perfect knowledge of the current internal state at some points in time separated by a bounded number of state changes.*

The observability property formally grants an adequate level of feedback meant to allow the correct internal state assessment by the human. Nevertheless the risk of information overload (providing too much information to the human [Los89]) is likely to jeopardize the correct human state assessment. Moreover some of the internal state variables are of no interest for the human.

[4] In the literature the concept of *mode confusion* has been widely discussed [BL02, LC99, Rus02, RCP99, BMPC98, LPS+97, JMH03] and it is somehow related to our conflict definition. However the authors' definitions of the mode confusion slightly differ between them. Moreover none of them totally matches with our conflict definition. Therefore to avoid any misunderstanding we will not use the term "mode confusion" in this work.

The approach presented in section 2.4.1 tries to overcome the limits of the simple observability verifying a kind of *observability that matters* called *full control*.

Other authors (see section 2.4.2) focus on the expert definition of undesirable characteristics of the internal state changes that may compromise the correct internal state assessment. In almost all those definitions[5] the observability issues are not taken into account. Indeed observability is *implicit* and should have been already verified. Those approaches are not interested in the question *has the relevant feedback been provided* but in *how the state change has been performed.*

Finally other authors (see section 2.4.3) focus on the formal evaluation of the consequences of assessment errors (defined by experts). Note that those assessment errors are neither deduced by an observability approach nor by a vulnerability definition. Indeed they should be defined *ad hoc* by the designers to describe a particular and well known case. Nevertheless the designers may represent errors due to observability issues too.

More details about some of those works are given in the following sections in order to take cues for our further developments.

2.4.1 Full control property

A property inspired by the observability but that tries to overcome the observability inadequacy is the *full control* [CGPF11]. The full control is defined as follows: *at each time during the interaction between the human and the machine, the human must know exactly what are the available commands on the machine and must be aware of at least the observations that can occur*.
The author calls *commands* the selection behaviours, *observations* the automated behaviours notified to the human *via* a feedback and *internal transitions* the automated behaviours that are not notified to the human at all.
The full control property verification is a necessary[6] condition for conflict prevention: the human should at least be aware if their selections has an effect and should at least expect all the possible feedbacks.
We propose two natural language interpretations of the full control property. The first one is: “the relevant (to know the available commands and being aware of the possible observations) part of the internal state must be observable”. The second one is: “the observable part of the internal state includes the relevant part of the

[5] Except the *no feedback* vulnerability.
[6] But not sufficient.

internal states".
Both interpretations have the same meaning but highlight two different approaches for the verification of the full control property. In the first case the design of the machine (including what the author calls commands) come first. Then the internal state is reduced to its relevant part and the interface feedback is designed to ensure the observability of the relevant part. The resulting H/M interface is as simple and compact as possible.
In the second case the full control property is tested on the existing design of the H/M interface. If the property is not verified the H/M interface is enhanced adding feedbacks. This approach could lead to more feedbacks than needed, resulting in information overload risk.

2.4.2 Vulnerabilities

Another approach that can be found in the literature for conflict prevention is the *vulnerabilities* analysis: indeed authors have defined a set of *vulnerabilities* defined as:

Vulnerability: *undesirable characteristics of internal state changes.*

Those authors neither use the term *conflict* nor define a concept similar to conflict. Indeed they somehow define it by *shadowing*: they define *undesirable* characteristics to be avoided (or just to be careful about, if it is not possible to avoid them). For us vulnerabilities are *undesirable* because they could bring about a *conflict*. The relation between vulnerabilities and conflicts is not deterministic: vulnerabilities just *prepare the ground for conflicts*.

We will call a *selection behaviour* the result of a *selection* (see figure 1.1 in chapter 1) on the internal state and on the H/M interface; an *automated behaviour* as a state change that is not fired by a selection with any possible relevant change on the H/M interface. We will use the term *behaviour* to define either a selection behaviour or an automated behaviour.
The more vulnerabilities a behaviour has, the more it will be the possible cause for conflicts.

Table 2.2 gives a detailed list of vulnerabilities. Note that different authors sometimes define the same vulnerability with a different name. For instance we put the *interface interpretation errors* [LPS$^+$97] and the *lack of appropriate feedback* [LPS$^+$97] in the same category, when the former focuses on the lack of observabil-

ity on the machine state and the latter focuses on feedbacks that do not inform on all the state changes implications, therefore leading to a state which is not totally known.
Some vulnerabilities are particular cases of other vulnerabilities, for instance the no feedback [CCH08] is an action triggered behaviour with no feedback, i.e. it is a special case of *lack of appropriate feedback* [LPS+97] with no feedback at all.
The *mode inconsistencies* [GTC05] is a special case of a *moded behaviour* [Fea05], in which one mode is used only rarely. More precisely for the *mode inconsistencies* a selection triggers one behaviour in most cases, but exceptionally it triggers another one. The greater the number of normal cases compared to exceptions the higher the assessment error risk. Note that this definition needs an arbitrary choice of a critical ratio between normal cases and exceptions.
The *operator authority limits* [LPS+97] is a special case of *inhibited behaviour* [Fea05] in which the inhibition is due to authority issues.
Note that a single behaviour may match many vulnerabilities at the same time. For instance an automated behaviour that is not communicated to the human matches the *automated behaviour* and *lack of appropriate feedback* vulnerabilities. A selection behaviour that in some cases exceptionally leads to a particular state with no feedback given to the human matches the *no feedback* and *mode inconsistencies* vulnerabilities.

Authors' definitions	Simplified description
inconsistent behaviour [LPS+97], moded behaviour [Fea05], no behavioural consistency [CCH08]	the same selection has many different consequences depending of the actual internal state
unintended side effects [LPS+97]	part of the selection behaviour is not intended by the human
mode inconsistencies [GTC05]	the effect of a selection on the internal state is almost all the time the same except for some special cases
indirect mode changes [LPS+97], automated behaviour, [Fea05]	behaviours that are fired without the need of a selection
operator authority limits [LPS+97]	some selections are not taken into account because the automation has the authority so as to satisfy some constraints
interface interpretation errors [LPS+97], lack of appropriate feedback [LPS+97], no feedback [CCH08]	different internal state values are represented on the interface *via* the same feedback
similar feedback [Vak00, Fea05]	the same display being used for more than one internal state change
armed behaviour [Fea05]	state change requested by a selection but automatically triggered
inhibited behaviour [Fea05]	some selections have no effect in some machine states
off nominal behaviour [PHB02, Fea05]	state changes that are executed in situations defined as off-nominal

Table 2.2: Vulnerable behaviours.

2.4.3 The comparison of a machine model and a mental model

One kind of approach for conflict prediction is the two-finite state machine (FSM) comparison. Rephrasing almost literately [Rus02]: if an FSM specification is available for the machine, and if we can construct one for a plausible FSM that the human thinks describes the machine behaviour (this FSM is called *mental model*), then we could, in principle, "run" the two FSMs in parallel to see if their behaviours ever diverge from one another. What is potentially valuable about this approach is that a body of techniques from the branch of formal methods in computer science known as "model checking" can be used to compare all possible behaviours of both FSMs.
The real issue for the application of this approach is the definition of the *mental model*: the designer should be able to *construct a mental model that a human might plausibly employ* [Rus02], but the issue to define a *plausible* mental model remains unsolved. Modifying the system state machine merging state values that could be considered "similar" from the human point of view is a possible way to define the mental model. However this kind of simplification corresponds to an *ad hoc human error* definition: the designer implicitly includes the causes of the conflict.
Nevertheless this method is well suited for a better understanding of the consequences of a well-know and frequent human error.
Model checking methods [BBS12] could also be used to check some properties like the reachability of an undesired state or the presence of deadlocks.

2.4.4 Conclusion: a comparison of approaches

It is worth noting that the previously described approaches differ not only in their practical implementation but also conceptually.
The *full control property* analysis is based on the formal verification of a necessary condition: the human should be able at least to know at any time the set of possible actions they are allowed to perform and be aware of the possible feedbacks they could receive *via* the interface. Nevertheless if the full control property is verified there is no certainty about the correct human situation assessment. If the full control property is not verified human-machine conflicts are highly probable[7], whereas the vulnerabilities analysis highlight what could possibly bring about conflicts.
The *vulnerabilities* analysis is based on an expert definition of the undesirable characteristics that may potentially lead to conflicts. Those characteristics are not systematically problematic: this approach highlights the situations for which *conflicts are likely to arise*.

[7]They should in principle arise eventually because the human does not know exactly what the available commands are on the machine or they are not aware of the observations that can occur.

Finally the *machine and mental model comparison* focuses on the assessment of the consequences of a well-know conflict case and is not meant to be a general approach, because it needs some arbitrary assumptions on the mental model.

In conclusion the approaches are complementary and focus on different aspects of conflict prevention: it is quite reasonable for a designer to test the full control property first (see in section 2.4.1), then to analyse the vulnerabilities (see in section 2.4.2) and possibly to make hypotheses on the mental model to evaluate the conflict dynamics (see in section 2.4.3).
In the frame of the knowledge conflicts due to the wrong human assessment of

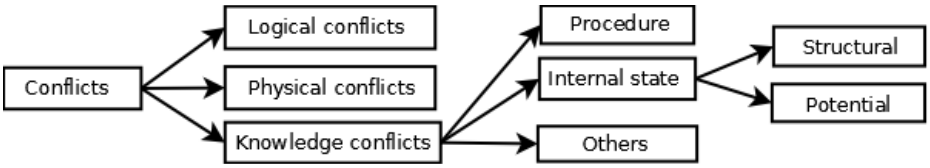


Figure 2.1: Partition of the conflict categories.

the internal state of the machine (or conflicts), we will call *structural conflicts* the conflicts due to a lack of observability of the internal state (see figure 2.1). They represent the conflicts the human incurs independently of a gap of their attention. We call *potential conflicts* the conflicts the human incurs because of a gap of their attention.
We will develop a model that can detect some structural conflicts (that have been proved to be relevant based on a literature survey), detect potential conflicts similar to those structural conflicts, and analyse the conflict dynamics. The model should be as general as possible.

A comparison of the reviewed methods and the approach we will propose (Target approach) is given in table 2.3.
Structural conflicts are detected by the approaches that take into account internal state observability issues. That is the case for the observability verification and the no feedback verification (among the vulnerabilities), so as for the full control property that verifies a relaxed version of the observability property[8]. The comparison of a machine model and a mental model detects all the structural conflicts in so far

[8]So it cannot detect particular situations of structural conflicts for machines (and relevant H/M interface) that verify the full control property. For instance consider the case of a machine that has no *observation* and allows all *commands* no matter the internal state: this machine behaviour verify the full control property, nevertheless errors in the internal state assessment will arise as soon as an *internal transitions* is fired.

as the mental model represents the actual feedback emission. For our approach the proof of the completeness of the structural conflicts detection is not given.
Potential conflicts are detected by the approaches that take into account the possibility that emitted feedbacks may be lost, because of their bad design of because of a gap in the human attention. So only the vulnerability analysis, the comparison of a machine model and a mental model (in so far as the mental model represents the lost of bad designed feedbacks) and our approach can detect them.
The conflict dynamics analysis (i.e. the evolution of the situation before and after the conflict arising) is possible for the finite state machine approaches, i.e. the comparison of a machine model and a mental model and our approach.
An approach is said to be general if it requires few assumptions about the nature of the human errors in the situation assessment. For the comparison of a machine model and a mental model the designer should be able to construct a mental model that a human might employ, so this approach is not meant to be general.
The information overload risk (i.e providing too much information to the human thus endangering the correct internal state assessment) concerns the observability verification and the full control property because they both verify a lower bound for the provided information[9]. Nevertheless all the approaches that can detect structural conflicts are somehow exposed to the information overload risk.

In the next chapter we propose a formal method to meet the objective to reduce conflicting situations *via* prevention.

[9]Note that for the full control property the lower bound to be verified is less demanding compared to the observability verification.

Method	**Structural conflict detection**	**Potential conflict detection**	**Conflict dynamics**	**General approach**	**Information overload risk**
Observability verification	Yes	No	No	Yes	Yes
No feedback	Yes	No	No	Yes	Yes
Full control property	Some	No	No	Yes	Yes
Vulnerabilities (except no feedback)	No	Yes	No	Yes	No
The comparison of a machine model and a mental model	Some	Yes	Yes	No	Yes
Target approach	Some	Yes	Yes	Yes	Yes

Table 2.3: Approaches comparison.

Chapter 3

Petri nets conflict patterns

3.1 Introduction

Formal studies have been carried out to identify crew-autopilot conflicts. Finite state automata [LT05] are generally used for modelling the autopilot since its behaviour is known, discrete and deterministic [CJR00, Jav02]. Some authors [LP97, BMPC98] have used this approach to examine the design of an autopilot and its tolerance to human error. [Rus02] has implemented finite state automata to detect inconsistencies between the behaviour of the autopilot and the human operator's mental model of the autopilot logic: he has shown [RCP99] that this approach could describe and predict the conflict between an MD-88 autopilot and the crew (detailed in section 3.2). Nevertheless this approach faces strong formal limits as it requires assumptions about mental models to represent the crew's imperfect knowledge about the autopilot logic [CJR00]. Therefore an alternative approach must be considered.

In this chapter we model two real cases of human-machine conflicts with Petri nets and we show how conflict states correspond to deadlocks in the Petri net (see section 3.2). A general conflict model is then proposed (see section 3.3). In chapter 4 these patterns will be used to *a priori* identify potential pilot-automation conflicts in the Petri net model of an autopilot system.

3.2 What the heck is it doing?

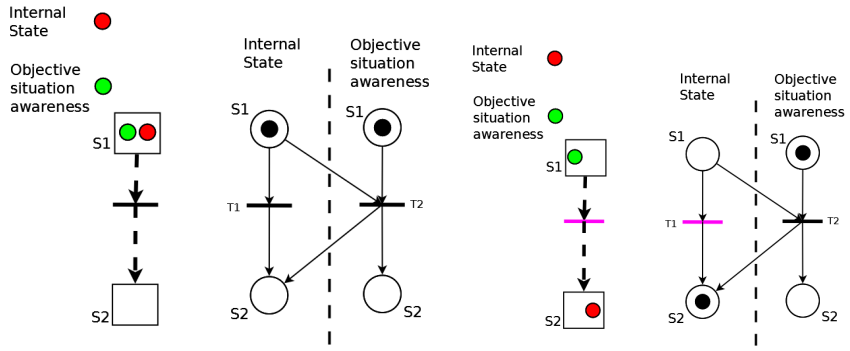
This section presents two real cases of human-machine conflicts. The first case (a *kill-the-capture* conflict with an MD-88 autopilot) has been reported by [Pal95] and investigated by [RCP99, Rus02]. The second case occurred during an experi-

ment campaign involving one of Onera's Ressac VTOL UAVs[1] in July 2011. For both cases we show that modelling the agents' possible actions enables the conflict to be identified in a formal way. Both cases will be modelled with Petri nets. In our Petri net representation of the human-machine system behaviour two state variables have an essential role. The first one is the *internal state*. The other one is the *objective situation awareness* (OSA) for a self aware human defined as the hypothetical situation awareness of a human aware of their actions, who knows the logic of the machine but who is not aware of automated state changes. At first the part of the Petri net dedicated to the OSA is built as a copy of the internal state. It is connected to the part dedicated to the internal state *via* transitions representing the exchanges of information between the human and the machine (feedbacks and selections). The part of the Petri net dedicated to the OSA may be enriched (adding places and/or transitions) to represent procedures the human has to follow.

3.2.1 A graphical convention

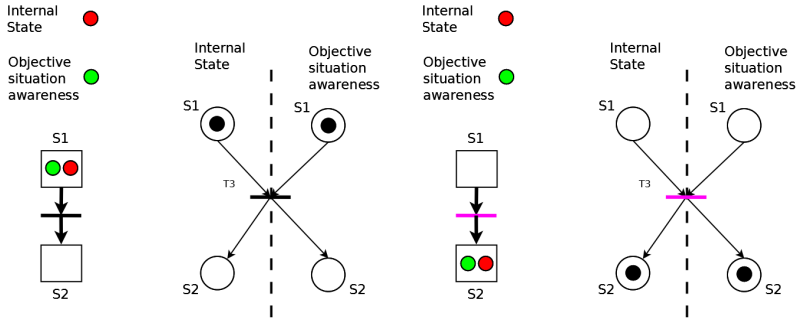
In the Petri nets representing the real cases we have studied, the internal state and the OSA may be connected in three possible ways. For the sake of graphical simplicity we define a graphical convention that is represented on the left in figures 3.1, 3.2 and 3.3 whereas the corresponding Petri net is represented on the right. Note that in this graphical convention an automated state change is represented by dashed arrows (see figure 3.1), and a synchronized state change by continuous arrows (see figure 3.2).
An automated state change with associated feedback is shown on figure 3.1a. The firing of transitions T1 or T2 represents the occurrence of an automated state change. If the feedback is emitted and correctly received it results in the firing of transition T2. If there is no feedback associated to this change or if the feedback is lost, this results in the firing of transition T1 and the final state is represented in figure 3.1b.
A transition that is fired for both the internal state and the OSA is shown on figure 3.2 (for instance this is the case for a selection state change): if properly initialized the evolution of the internal state and the OSA are synchronized.
The effects of a state change that is specified by the procedure the human has to follow but does not correspond to an actual change in the internal state is represented on figure 3.3. Note that in the graphical convention events may fire many transition at the same time (figure 3.4).

[1] Vertical Take-Off and Landing Unmanned Aerial Vehicles



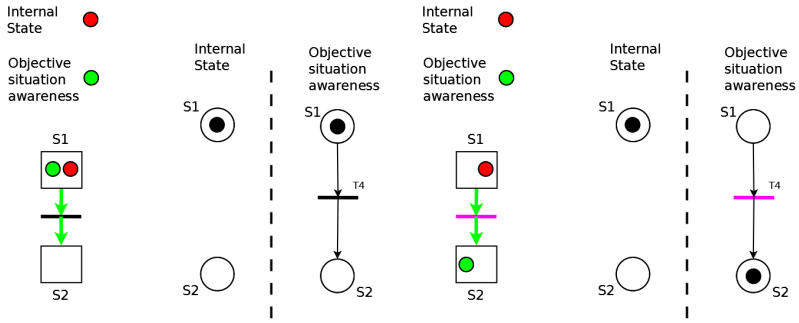
(a) Initial state. (b) Final state if feedback lost.

Figure 3.1: Automated state change.



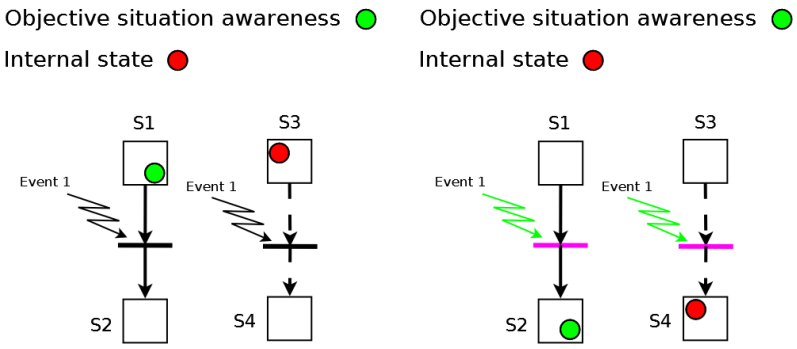
(a) Initial state. (b) Final state.

Figure 3.2: Synchronized state change.



(a) Initial state. (b) Final state.

Figure 3.3: State change for the OSA only.



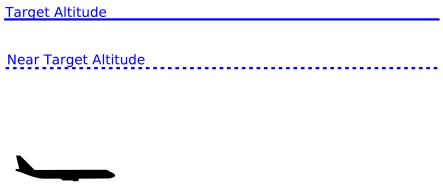
(a) Initial state. (b) Final state.

Figure 3.4: Event effect in the graphical convention.

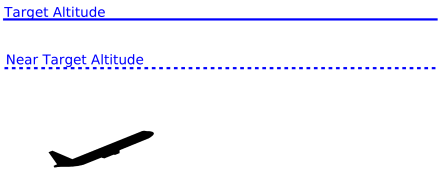
3.2.2 The kill-the-capture conflict

The two agents involved are the Autopilot of the MD-88 and the Pilot. The transitions that are considered are state changes (i.e. selection state changes and automated state changes). For the sake of clarity only the relevant modes and mode transitions are represented. Unlike Rushby [Rus02], we do not make any specific assumption about a *mental model* of the Pilot, but we take the viewpoint of what the Pilot actually does. In the Petri nets, we use the same colour code as in [RCP99]: green for done by the Pilot, red for done by the Autopilot. In figures 3.6 to 3.9 the Petri net representation is given for the initial and final states, whereas the graphical convention is used after each transition firing. In figure 3.5 a schematic representation of the situation is provided too.

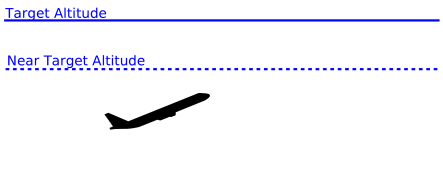
In the Initial state *Alt-Capture* mode of the Autopilot is not armed (initial marking "No Alt mode armed") – figure 3.6 (and see figure 3.5a).
The Pilot sets altitude to a target value, this causes Autopilot *Alt–Capture* mode to *Arm*, therefore the target altitude set by the Pilot will not be overshot. The Pilot also sets Pitch mode to *VSPD* (Vertical Speed – aircraft climbs at constant rate) figure 3.7a (and see figures 3.5b, 3.5c). The Pilot then sets Pitch mode to *IAS* (Indicated Air Speed – climb rate adjusted, constant air speed) – figure 3.7b.
When target altitude is nearly reached (see figure 3.5d), the Autopilot changes Pitch mode to *Alt Cap* (provides smooth levelling off at the desired altitude) for the actual internal state but not for the objective situation awareness: therefore for the internal state, mode *Alt Cap* is disarmed , so as *Pitch mode IAS* – figure 3.8a. The Pilot then changes Pitch mode to *VSPD*, therefore *Pitch mode Alt Cap* is disarmed for the internal state – figure 3.8b.
When event *target altitude* occurs, state Pitch mode *Alt Hold* cannot be reached since neither possible precondition is true (Alt capture armed or Pitch mode Alt Cap). Therefore event *target altitude* is "lost" and the aircraft goes on climbing at the *VSPD* indicated by the pilot, – figure 3.9 (and see figures 3.5e, 3.5f).
The "Oops, it didn't arm" uttered by the pilot reveals that he does not understand why the aircraft goes on climbing. In fact, his actions on the Autopilot modes have destroyed the Autopilot sequence. Formally the Petri net is blocked (i.e. no transition can be fired anymore since the human does not perform other selections until the attainment of Pitch mode Alt Hold). This is a *conflict* [TMFC00] as the OSA and the internal state are not coherent and this matters for the next steps of the mission: indeed the aircraft goes on climbing and is likely to violate separation constraints.



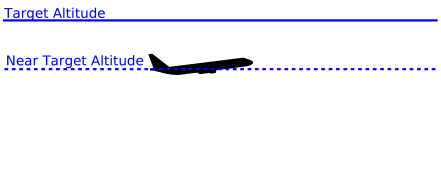
(a) Initial state



(b) Climb

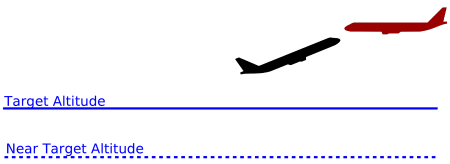


(c) Climb



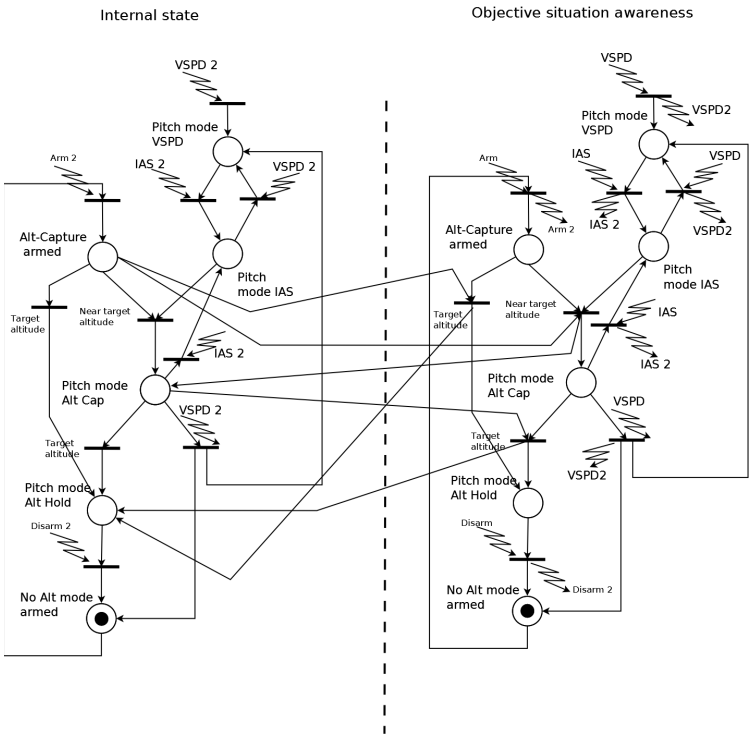
(d) Target altitude nearly reached

(e) The aircraft goes on climbing

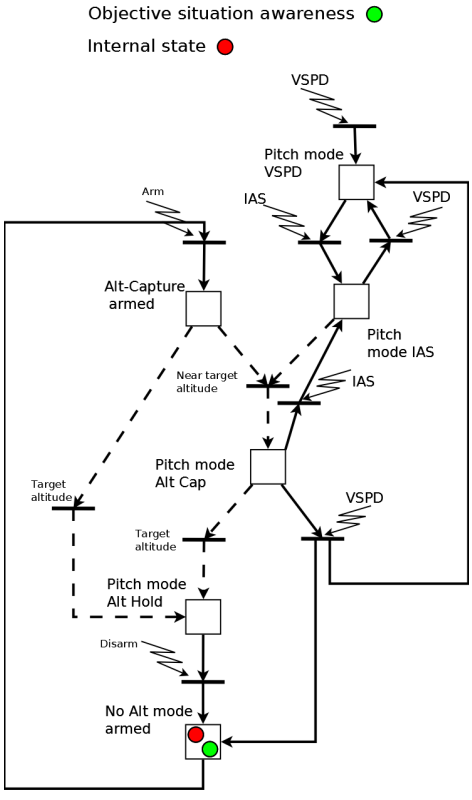


(f) The aircraft goes on climbing

Figure 3.5: Schematic representation of the situation

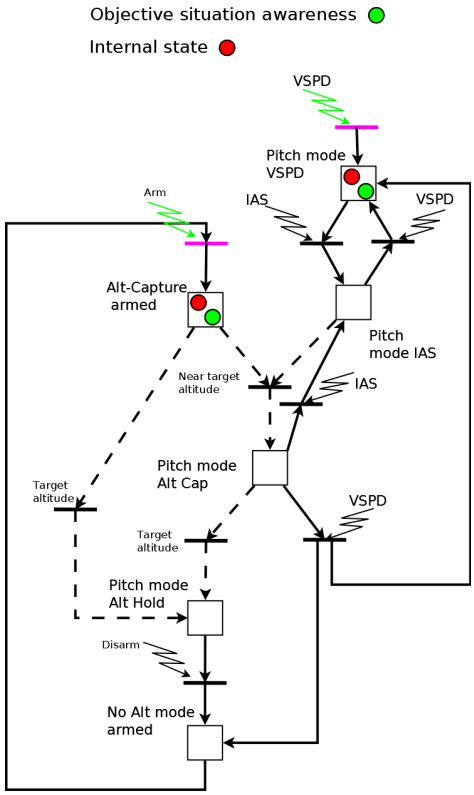


(a) Petri net

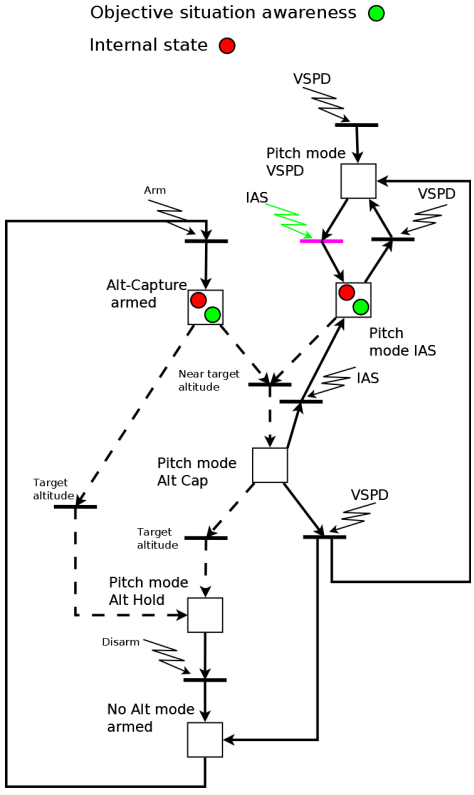


(b) Graphical convention

Figure 3.6: Alt-Capture not Armed

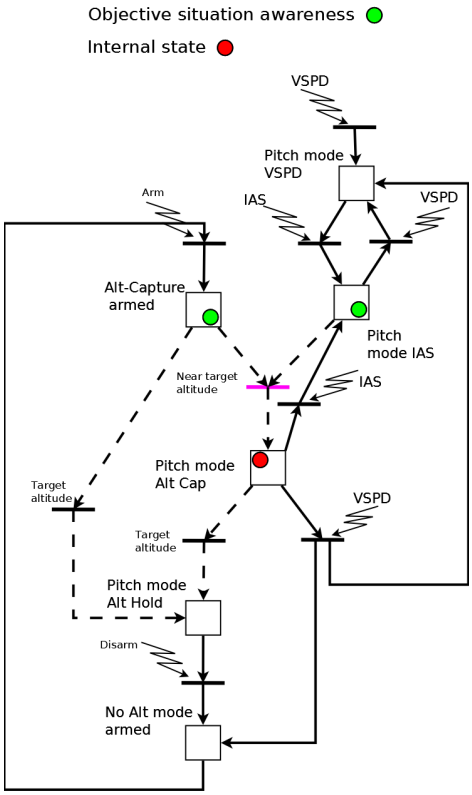


(a) Alt-Capture armed and VSPD armed

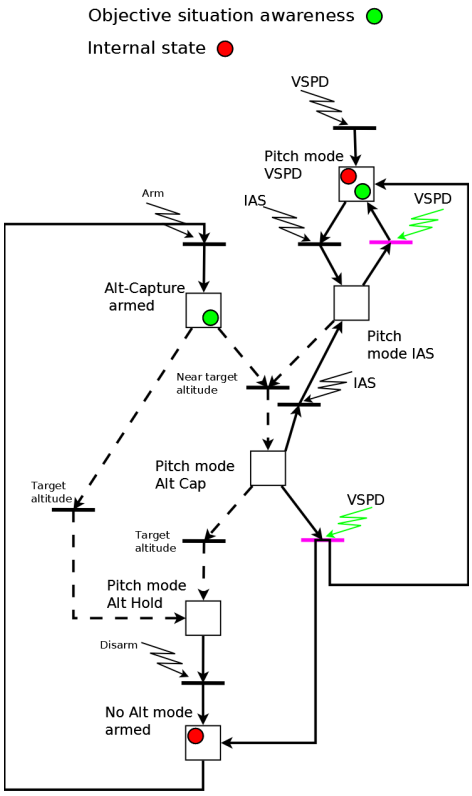


(b) IAS armed

Figure 3.7

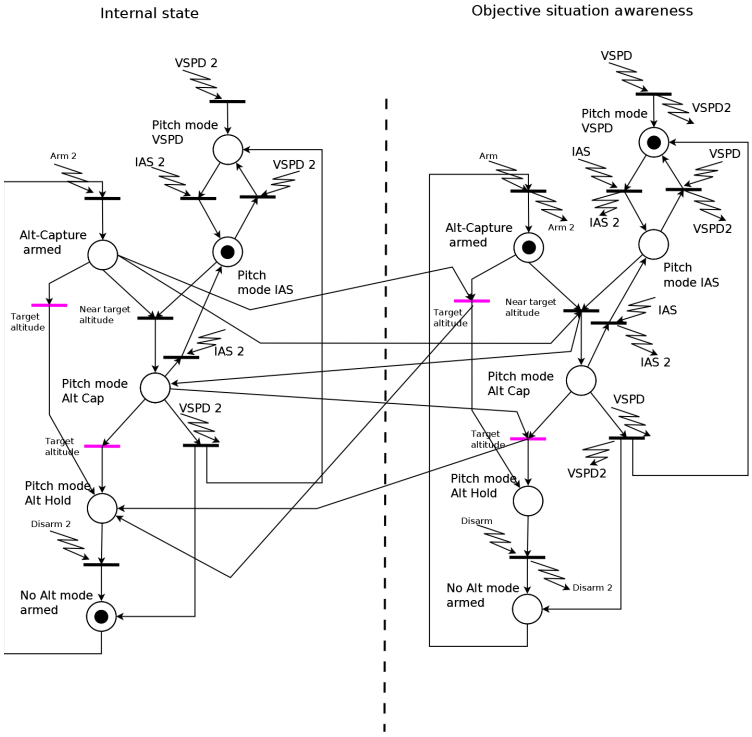


(a) Near target altitude

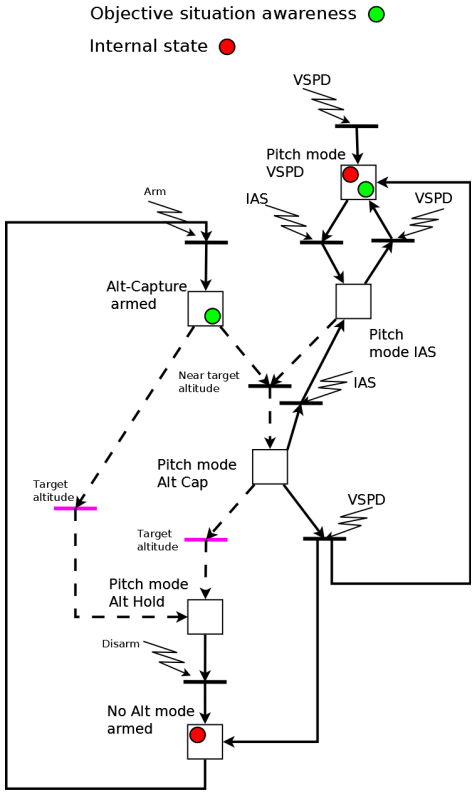


(b) Pitch mode VSPD

Figure 3.8



(a) Petri net



(b) Graphic convention

Figure 3.9: Event target altitude lost – “Oops, it didn’t arm” [Pal95].

3.2.3 Rain and automation

The second case of conflict occurred by chance during an experiment involving an Onera Ressac VTOL UAV in July 2011. Indeed the experiment was meant to test some properties of the Ressac planner and was not an ad-hoc scenario to bring about conflicts. The UAV mission requires two nominal pilots: the ground control station pilot (Gp) and the field pilot (Fp). For regulatory issues a third operator, the security pilot (Sp), can take over the manual piloting (as long as he wants) to deal with any unexpected event. About a dozen of other members of the Ressac team were checking the mission plan execution and performing other tasks.
There are five piloting modes (cf Table 3.1), one is totally automated (Nominal autopiloting- Autonav), three are partially automated modes and have been developed by Onera (Nominal autopiloting- Operator flight plan, Nominal manual- high level, Nominal manual- assisted), and the last one is a direct piloting mode (Emergency manual) using the on-the-shelf equipment of the vehicle (Yamaha RMax). The latter mode can be engaged only by the Safety pilot who has always preemption rights through activating an exclusion switch cutting off the machine commands. The Safety pilot communicates verbally the activation of the exclusion switch to the Fp and to the Gp. Notice that the Ressac software architecture has no visibility on the state of the switch. Flight phase transitions are allowed only in Nominal autopiloting mode.

	Automation	**Gp**	**Fp**	**Sp**	**Phase transitions**
Nominal autopiloting- Autonav	*				*
Nominal autopiloting- Operator flight plan	*	*	*		*
Nominal Manual- high level	*		*		
Nominal Manual- assisted	*		*		
Emergency Manual				*	

Table 3.1: Piloting modes, agents' involvement and phase achievement

So two nominal modes are possible i.e. Nominal autopiloting and Nominal manual piloting. When Nominal autopiloting is engaged, Ressac flies autonomously according to its plan, i.e. for this particular experiment:

- Phase 1: heading from the initial position to waypoint alpha
- Phase 2: heading from waypoint alpha to waypoint beta
- Phase 3: heading from waypoint beta to waypoint gamma

The following Petri nets represent the internal state values and state changes of the Ressac software agent (right) and of the Gp's objective situation awareness

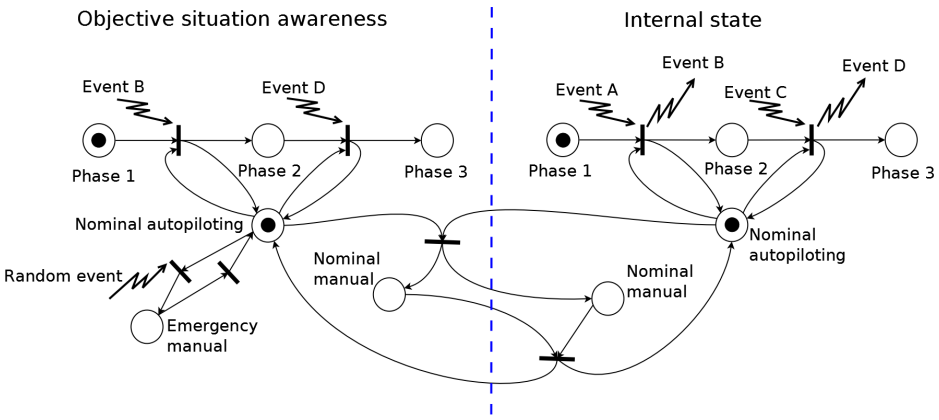


Figure 3.10: Initial state

inferred by the feedbacks, selections and the communication with the Sp[2] (left).
The OSA part of the Petri net (see figure 3.10 left) matches the structure of internal state part of the Petri net (see figure 3.10 right) except the fact that it includes the case of the Sp taking control of Ressac to deal with an emergency: in that case the OSA is that the mission is stopped. Initial state is human and machine both in state *Phase 1*.
In the Nominal autopiloting configuration the occurrence of *Event A* (waypoint alpha reached by Ressac) fires transition *Phase 1/Phase 2* for the internal state. This transition emits *Event B* (relevant change in the UAV flight direction and information displayed on the Gp H/M interface) which updates the OSA.
Transition *Phase 2/Phase 3* operates the same way with *Events C* (waypoint beta) and *D*.

What happened in July 2011 is the following sequence: Ressac was flying *Phase 1* heading for waypoint alpha, when it began to rain. This random event made the Safety pilot Sp take over the control on Ressac. On the Petri net of figure 3.11 transition *Random event* is fired (because of the decision taken by the Sp and relevant communication to the Gp) and *Emergency manual* place is marked.

While operating Ressac manually in order to make it land, the Sp unintentionally flew it over waypoint alpha. Therefore *Event A* is generated, and the software agent engages *Phase 2* (figure 3.12). *Event B* is lost on the OSA side, since one

[2]The Gp verbal communication with the Sp corresponds to the arc labelled as “other observations” in the Introduction, figure 1.1.

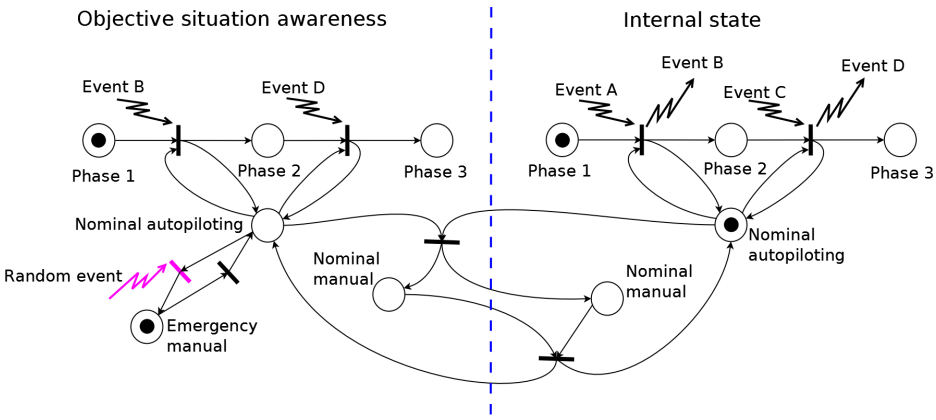


Figure 3.11: Rain and emergency manual mode

precondition (Nominal autopiloting) is no longer verified (figure 3.13).

Then the rain stopped and the Sp decided that the nominal plan could be resumed. Transition *Emergency manual* to *Nominal autopiloting* is fired (figure 3.13). The nominal plan was resumed (*Phase 2*) and Ressac headed waypoint beta. The Gp, who was expecting *Phase 1* to be resumed, did not understand what Ressac was doing and began to panic (as the Fp and the Sp). This is again a *conflict* [TMFC00] in which the human considered the behaviour of the machine as a failure and aborted the mission. Indeed none of the dozen test team members properly interpreted the behaviour of Ressac.

Notice that the marking of the Petri net (figure 3.13) is such that place *Phase 2* is marked on the internal state side whereas place *Phase 1* is marked on the OSA side. Nevertheless it is a matter of semantic inconsistencies and not of formal inconsistencies within the Petri net model.
Identifying conflicts through semantic inconsistencies would involve an explicit enumeration of all possible inconsistencies, which is hardly possible. Therefore what is relevant here from a formal point of view is not the semantic inconsistencies but the fact that the human agent part of the Petri net model is blocked (*Event B* will never occur again and *Phase 2* will never be marked).

The next section will focus on a generalization of agent conflict representation and detection.

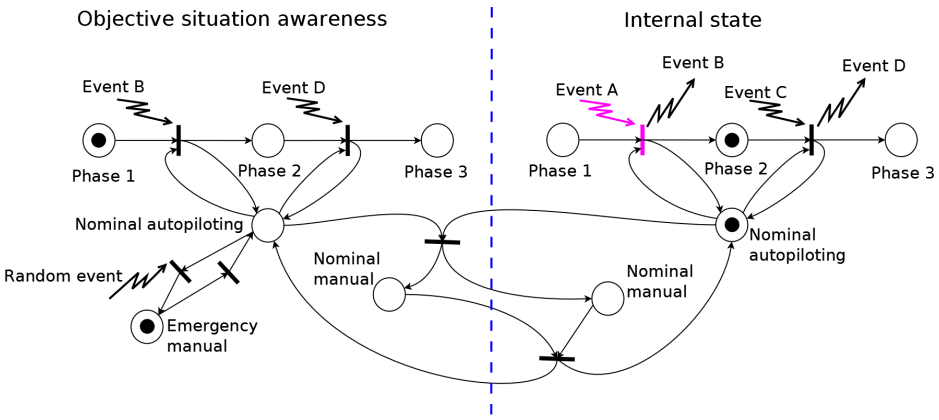


Figure 3.12: Software state update

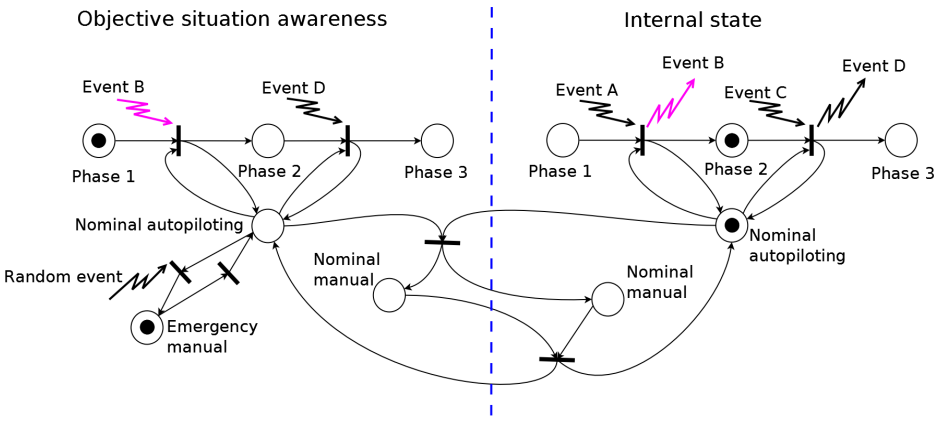


Figure 3.13: No update on the OSA side.

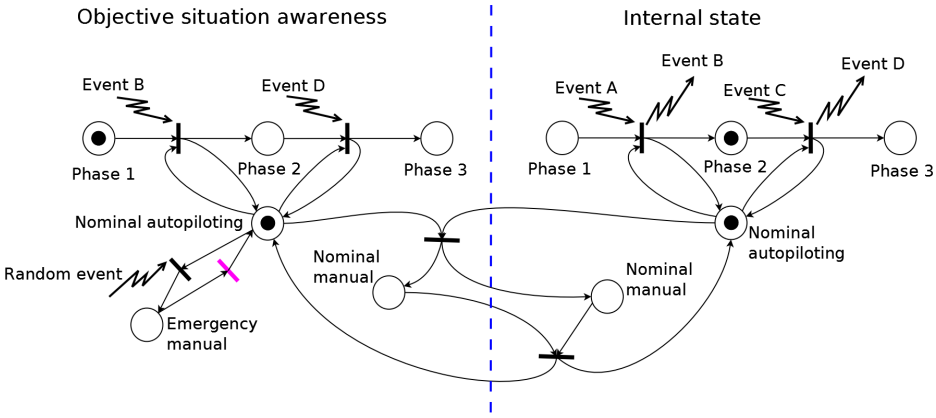


Figure 3.14: What the heck is it doing?

3.3 Towards generic patterns of human-machine conflicts

Automatic transitions may be notified to the human (*via* a visual or aural feedback) or may be "hidden". In the case of a poorly designed feedback, or because of a gap in the attention of the human, the feedback is likely to be missed: the transition is "unseen". On the other hand the structure of the internal state may be not compliant with the operational procedure the human has to follow: the effect of some events may be "hidden" from their point of view.
Considering real cases of "hidden" or "unseen" state changes, including those shown in section 3.2, we have modelled them with Petri nets [PSD12] in order to assess their consequences: firing such transitions in the Petri nets may lead to deadlocks and we have noticed from the real cases that the deadlocks correspond to conflicting situations. Therefore our objective in this section is to propose a generic pattern to identify these "hidden and blocking transitions" in human-machine system models.

3.3.1 The basic conflict pattern

In our automated changes analysis we assume that the human knows about the logic of the machine and the meaning of the feedbacks: if the human is "aware" of a state change from S1 to S2, they will actually believe that the resulting internal state is S2.
As previously mentioned in section 1.2, a state change can be a selection change or an automated change. Selection state changes will affect both the internal state

(the machine is aware of the selections because they affect the internal state) and the OSA (the human is aware of their own selections). Let us consider the selection change in figure 3.15. The internal state values are successively S1 then S2. In figure 3.15a, the internal state is S1 and both agents' (human and machine) knowledge is the same. The result of the firing of transitions T1 and T2 is that both agents' knowledge about the machine internal state is S2 (figure 3.15b). The next paragraph deals with less symmetrical cases corresponding to human-machine interaction poor designs that may result in a conflict.

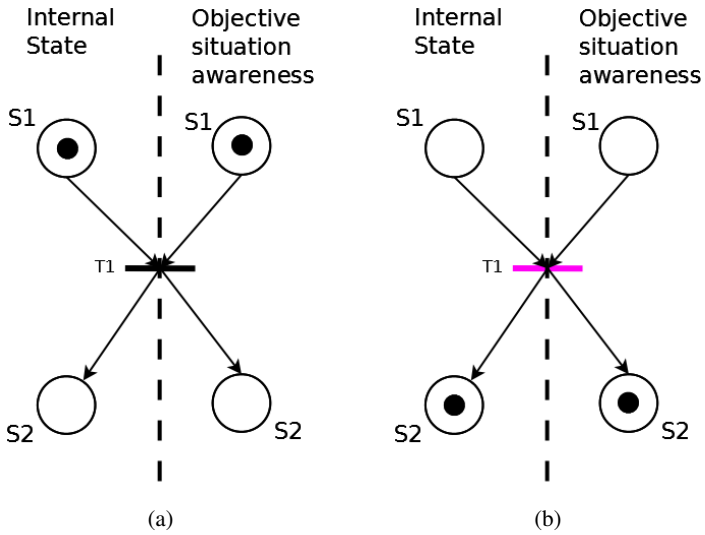


Figure 3.15: The internal state and the objective situation awareness are the same.

Let us consider a scenario in which two transitions T1 and T2 can be fired (see figure 3.16). The firing of transition T2 (i.e. a "perceived" automated change) makes both the internal state and the human's OSA evolve from state S1 to S2 (see figure 3.16b). On the contrary (see figure 3.16c) the firing of transition T1 only makes the internal state evolve to S2 whereas the human's OSA is still S1.

The case of a lack of feedback for T1 (i.e. "an hidden" transition) is a structural deficiency whereas a loss of feedback associated with T1 (i.e. "unseen" transition) is a potential vulnerability.

Semantically both cases amount to the same conflict: the human is not aware of an automated change. Nevertheless this is a matter of semantic inconsistency and not of formal inconsistency within the Petri net model. Therefore what is relevant here from a formal point of view is not the semantic inconsistency (i.e. the internal

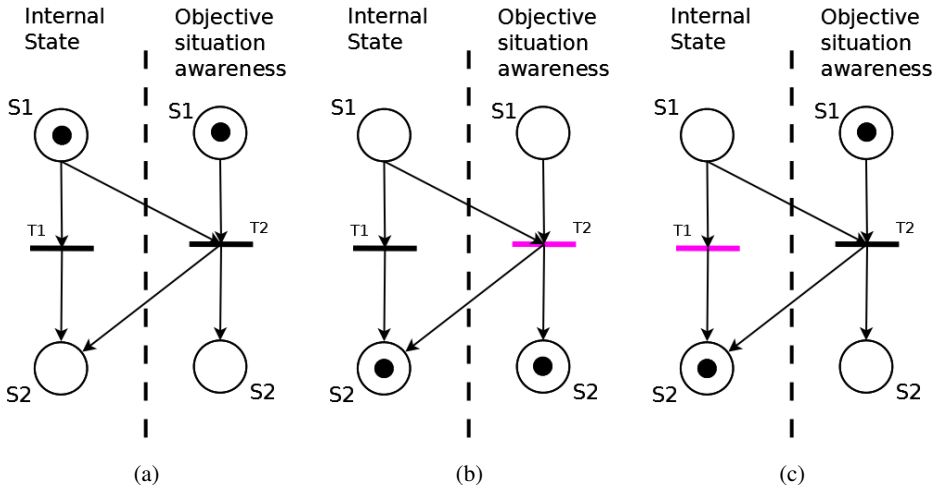


Figure 3.16: An automated change.

state is S2 whereas the human's OSA is S1) but the fact that transition T2 is dead: it cannot be triggered anymore since there is no token anymore in one of its input places.

Other conflict patterns may be derived as variants of this one: the keypoint is the fact that there is a state change that affects only the internal state or the OSA: for instance an automated "hidden" or "unseen" state change. Three other pattern variants are presented hereafter.

Human authority limits

This case has been described by [LPS+97], the author calls it *operator authority limits*. This case is among the vulnerabilities shown in section 2.4.2.
The firing of transition T2 (in this case a selection change) makes the internal state and the OSA evolve to S2 (figure 3.17b). Nevertheless the firing of the "hidden" transition T1 (automated state change) results in different states for the internal state (S1) and the OSA (S2). Semantically, this is a conflict: the selection has been nullified by the machine, and the human is not aware of that. Structurally transition T2 is dead (figure 3.16c).

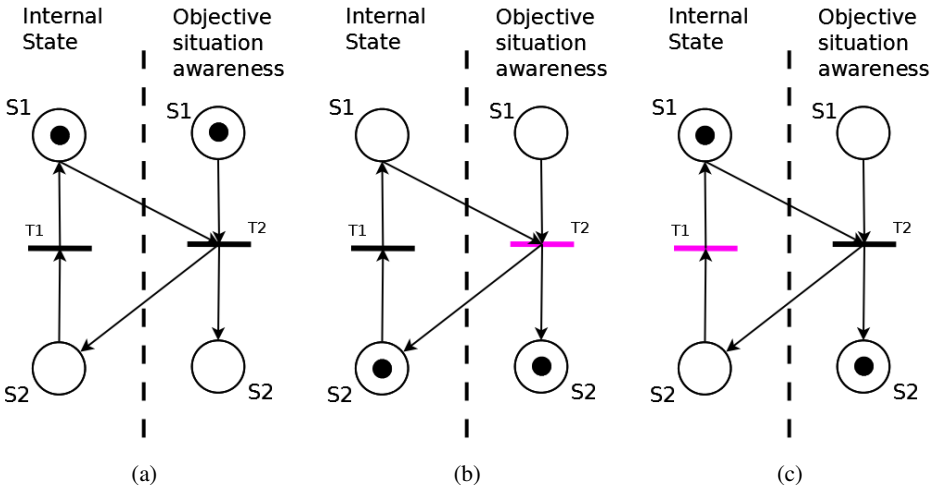


Figure 3.17: An automated change nullifying a selection change.

Example Let us consider the case of an aircraft controlled by a pilot and an autopilot. At first the pilot is manually controlling the aircraft (autopilot Off) and the flight speed is slightly above the maximum flight speed. A pilot selection meant to connect the autopilot will have as an effect a temporary connection of the autopilot (autopilot On) followed by an automatic disconnection reversing the effect of the selection[3].

[3]This example is just meant as a didactic one and does not necessary match with a real autopilot behaviour.

Side effect

This case has been described by [LPS+97], the author calls it *unintended side effect*. This case is among the vulnerabilities shown in section 2.4.2.
The firing of transition T1 (in this case a selection change) makes the internal state and the OSA evolve to S2 for variable A, and the internal state to S4 for variable B. The result is that the OSA for variable B is S3 whereas the internal state is S4. Semantically this is a conflict: the human is not aware of a side effect of a selection. Structurally, transition T2 is dead (figure 3.18b[4]).

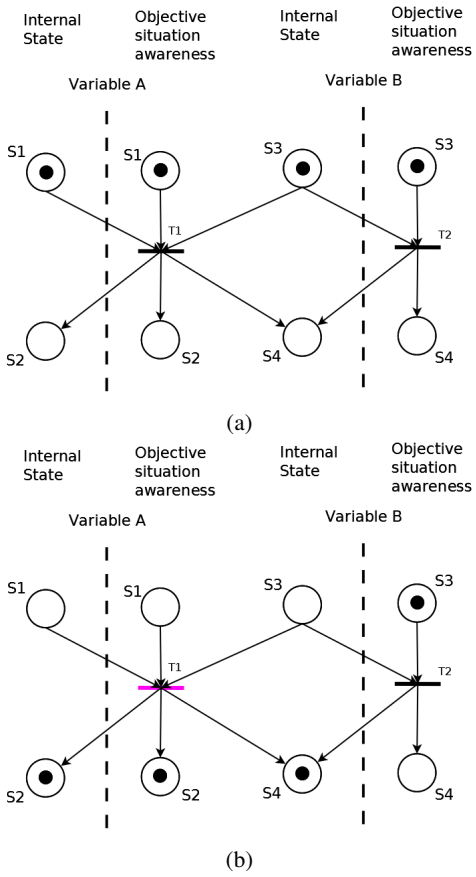


Figure 3.18: A state change represented only for the OSA

[4]For the sake of simplicity in figure 3.18 the state transition corresponding to the correct uptate of both the OSA and the internal state is not represented.

Example Let us consider the case of an aircraft controlled by a pilot and an autopilot. At first the autopilot is controlling the aircraft (autopilot On), the vertical mode is Climb and the lateral mode is Navigation. Subsequently the pilot changes the lateral mode from Navigation to Heading. As a side effect the vertical mode changes too, from Climb to Open climb.

State change represented only for the OSA

The firing of transition T1 makes the OSA only evolve from S1 to S2. The result is that the OSA evolve to S2 whereas the internal state is still S1 (figure 3.19b).

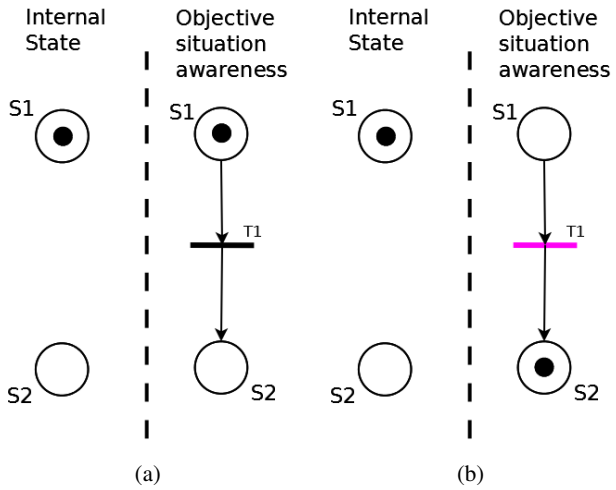


Figure 3.19: A state change represented only for the OSA

Example In the rain and automation case (section 3.2.3) the transition to *emergency manual* only exists for on the OSA side.

3.4 Conclusion

Obviously this formal approach cannot predict the effect of these potential conflicts on the human's behaviour. If a feedback is emitted (no matter how bad its design is) the human could correctly receive it and understand the relevant internal state change. Moreover without the need for the feedback to be received[5] the human

[5]Because it is not emitted or because it is not perceived.

could still correctly assess the internal state observing the resulting machine "behaviour". Therefore experiments have to be conducted in order to assess the actual human's behaviour when facing the *a priori* detected critical transitions. Chapter 4 focuses on an experiment we have designed in order to test the soundness of the formal approach, in other words to assess whether the *a priori* identified patterns indeed create conflicts between the automation and the pilot, and whether those conflicts are detected or solved.

Chapter 4

A Flight Simulator Experiment

In order to test the soundness of the patterns that have been proposed in chapter 3 to detect conflicts in human-machine interaction models, we have designed an autopilot system and modelled it with Petri nets. In section 4.1 the patterns are used to identify potential pilot-automation conflicts in the model of the autopilot system. Next general aviation pilots were placed in our flight simulator equipped with the same autopilot and had to deal with these different potential conflicting situations (see section 4.2). Their flight performance and post experiment debriefing were used to assess whether these situations actually led to conflicts with the automation, and whether they were detected or not, and solved or not.

4.1 Pattern identification in an autopilot model

The objective of this chapter is to provide a first validation of the patterns on a concrete use-case. Therefore we designed an autopilot system and modelled it with Petri nets. Pattern identification was performed on this model through the detection of dead transitions and pattern matching. Then these patterns were tested with general aviation pilots in our flight simulator in order to assess whether they actually brought about conflict (see section 4.2).

4.1.1 Autopilot logic

The logic of the autopilot we designed is inspired from different modern autopilot systems. It works as follows:

The autopilot is engaged and disengaged *via* a push button “AP” on Flight Control Unit. The disconnection of the autopilot is accompanied by a “cavalry

charge" auditory warning. The autopilot must be disconnected to fly manually.

The lateral trajectory ("heading" mode) is controlled *via* a dedicated knob on the the Flight Control Unit. The vertical trajectory ("vertical speed" mode) is controlled *via* two knobs to program respectively the target altitude (e.g. 6000 ft) and the vertical speed (Vz) that is either positive (e.g. +2000 ft/mn) or negative (e.g. -2000 ft/mn). The vertical speed is zero (i.e. 0 ft/mn) when the autopilot reaches the target altitude or when the pilot pushes the vertical speed knob to level off. The different flight modes (heading, vertical speed) are displayed on the upper part of the Primary Flight Display.

Near overspeed mode reversion: if the speed is 5 knots close to the maximum speed (Vmax) and the vertical speed is zero or negative, then the autopilot climbs at +1000 ft/mn to anticipate a possible overspeed. The new vertical speed is displayed on the upper part of the Primary Flight Display.

Autopilot automatic disconnection: if the aircraft exits the flight envelope (maximum speed or low speed /stall), then the autopilot automatically disconnects. In this situation, the priority speed auditory warning ("triple chime") is triggered and inhibits the autopilot disconnection warning. "AP" indicator disappears from the Primary Flight Display to indicate the autopilot disconnection visually.

Inconsistent programing: if the target altitude is inconsistent with the selected vertical speed (e.g. target altitude greater than the current altitude and negative selected vertical speed) then the autopilot levels off the airplane.

4.1.2 Pattern identification

From the previously described logic, we have modelled the interactions between the pilot and the autopilot with Petri nets (see figure 4.1). The result of the formal analysis of those Petri nets is that transitions T1, T2 and T3 can be dead. The pattern matching with the previously described patterns shows that they correspond to three instances of the generic conflict pattern (dashed line boxes). Hereafter we explain the three of them precisely.

Near overspeed mode reversion (T1) Because of the pilot speed selection (or also because of strong tail winds), the aircraft may fly near the maximal speed (Vmax - 5 knots). Initially the aircraft is levelling off (in figure 4.1 Speed: Normal; Vertical Speed: level off; Autopilot connection: ON; for both autopilot and pilot).

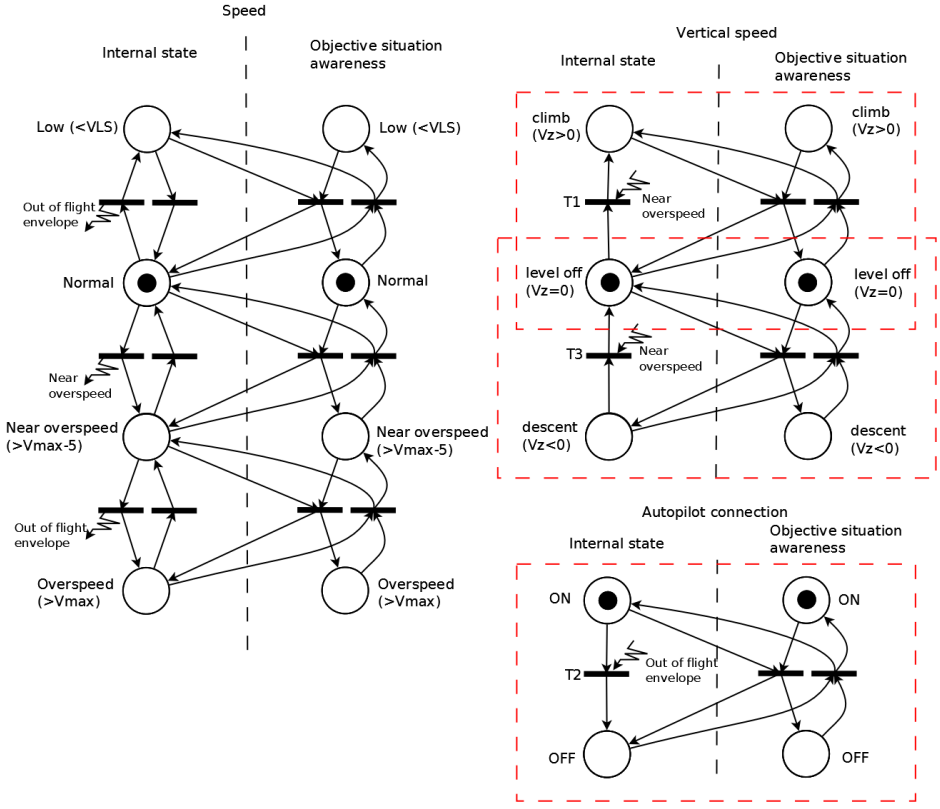


Figure 4.1: The Petri net model of the autopilot logic. The boxed parts are the generic conflict pattern instances associated with the dead transitions.

If the near overspeed event is fired (Speed from Normal to Near Overspeed for the autopilot) the autopilot pitches up the aircraft (in order to slow it): transition T1 is fired and the new state is Vertical Speed: climb for the autopilot and Vertical Speed: level off for the pilot. The transition to this mode is notified to the pilot by a change in the Primary Flight Display: the actual vertical speed is shown in red and is no longer coherent with the vertical speed selected on the Flight Control Unit. Nevertheless, because of the poor feedback design, we assume that the notification of the mode transition will not be perceived by the pilot.

Near overspeed mode reversion (T2) Usually an aircraft flies near the maximal speed with the autopilot engaged (ON) (in Figure 4.1 Speed: Near Overspeed; Autopilot connection: ON; for both autopilot and pilot). In case of strong tail winds, the aircraft accelerates and may exceed the maximal speed limit (event out of flight envelope), which leads to automatically disconnect the autopilot (transition T2 is fired and the new state is Autopilot connection: OFF for the autopilot and Autopilot connection: ON for the pilot). An overspeed auditory alarm is triggered and the autopilot status on both the Primary Flight Display and the Flight Control Unit is changed. As two auditory alerts cannot be broadcast at the same time, the autopilot disconnection alarm is inhibited.

Near overspeed mode reversion with inconsistent programing (T3) Initially the aircraft is descending (in Figure 4.1 Speed: Normal; Vertical Speed: descent; Autopilot connection: ON; for both autopilot and pilot). If the near overspeed event occurs (Speed from Normal to Near Overspeed for the autopilot) the near overspeed mode reversion should change the vertical speed from descent to climb, but because the target altitude (which is lower than the current altitude) is inconsistent with this vertical speed the autopilot levels off the airplane. For the sake of simplicity the combined effects of both off-nominal behaviours is directly shown in the Petri net: transition T3 is fired and the new state is Vertical Speed: level off for the autopilot and Vertical Speed: descent for the pilot. Information is sent about the triggering of the protection on the Primary Flight Display.

Note that T1, T2 and T3 are the only transitions that lead to a deadlock in the Petri net with certainty: it is not the case for all the automated transitions on the Speed (see figure 4.1, left).

4.2 Flight Simulator Experiments

4.2.1 Hypothesis

This section focuses on the experiment we have designed in order to test the soundness of the formal approach, in other words to assess whether the *a priori* identified patterns indeed create conflicts between the automation and the pilot, and whether those conflicts are detected or solved. Therefore the experiment is based on a scenario involving the three patterns that have been identified with the formal approach. Our hypothesis is that the mode transitions that have been identified *a priori* as vulnerabilities with the formal approach indeed generate human-machine conflicts in the experimental scenarios, and that these conflicts may remain unnoticed or unsolved by some participants.

4.2.2 Participants

Ten healthy volunteers (one female; mean age = 38.2 year, SD = 16.3; mean flight experience = 2997.8 hours, range = 55 – 12000; automated flight desk experience = 314,4 hours, range = 10 - 1000), all French defense staff from Institut Supérieur de l'Aéronautique et de l'Espace (ISAE) campus, were recruited by local advertisement. The participants gave their informed consent after receiving complete information about the nature of the experiment.

4.2.3 Material: flight simulator

The ISAE 3-axis motion flight simulator was used to conduct the experiment (see figure 4.2). It simulates a twin-engine aircraft flight model and reproduces aerodynamic effects such as buffeting (i.e., aircraft vibration during stall). The user interface is composed of a Primary Flight Display, a Navigation Display, and the upper Electronic Central Aircraft Monitoring Display page. The pilot has a stick to control the flight, rudder pedals, two thrust levers and a Flight Control Unit to interact with the autopilot. Two stereophonic speakers located under the displays on each side of the cabin were used to broadcast a continuous engine sound (77dB), and to trigger the alarms ("cavalry charge" – autopilot disconnection, "triple chime" – overspeed and stall) presented at 86.3dB, that is 8.5 times louder than the global ambient cockpit sound. For this experiment an autopilot (automatic control of the lateral and vertical trajectory) and an autothrust (automatic control of the thrust/speed) were designed. The logic of the autopilot has been described in section 4.1.1. As for the autothrust, the logic is as follows:

The autothrust is engaged and disengaged via a push button "ATHR" on the Flight

Figure 4.2: The ISAE 3-axis flight simulator. Left: outside view of the cabin. Right: cockpit view, the arrow indicates the position of the Flight Control Unit (FCU) dedicated to autopilot programing (heading, speed, altitude and vertical speed selection). The participants flew the aircraft from the left seat.

Control Unit. The autothrust must be disconnected to adapt the thrust manually. The states of the autothrust are displayed on the upper part of the Primary Flight Display.

4.2.4 Experimental scenario

The scenario that was proposed to the participants included the three situations that had been identified as problematic through the formal analysis (see section 4.1.2). The initial position of the aircraft was on the 14R runway at Blagnac airport (Toulouse, France). The ATC (Air Traffic Control) cleared the aircraft for takeoff and instructed the plane to "climb 3000 ft, 1500 ft/mn". When the aircraft reached 1000 ft, autopilot engaged, the ATC gave clearance to steer 330° and to increase speed up to 176 knots.

Situation 1 At 1200 ft, the ATC requested an immediate "level off to avoid an incoming aircraft". This situation led to the first situation: the aircraft started to level off but as the speed reached 176 knots (i.e. 5 knots below maximum takeoff speed) the autopilot climbed at +1000 ft/mn to anticipate possible overspeed (see section 4.1.1). This could potentially lead to a collision with the incoming aircraft as the aircraft climbed instead of levelling off as initially required.

Situation 2 When the aircraft reached 1500 ft, it was cleared to "climb 11000ft, 1500 ft/mn, steer 322°". At 10700 ft, the aircraft faced a jet stream leading to a very brief overspeed. This situation led to the second situation, i.e. the brief overspeed provoked the disconnection of the autopilot: the altitude capture at 11000

feet could not be achieved anymore and the aircraft kept on climbing with a risk of level bust.

Situation 3 Five minutes later, the ATC instructed the participant to "slow down, speed 300 knots, descend 5000 ft, 1000 ft/mn". At 9000 ft, the ATC required to "Accelerate, 325 kts, heading 140°". This situation led to the third situation: as the aircraft was descending, the speed reached 325 knots (i.e. 5 knots below maximum cruise speed) and the autopilot reversed to +1000 ft/mn. This led to an inconsistency as the selected target altitude, which was below the current altitude, could not be reached with a positive vertical speed. As a matter of fact, the plane levelled off (see section 4.1.1), instead of descending as initially requested.

4.2.5 Procedure

The participants were told about the real purpose of the experiment, i.e. that they would face off-nominal situations. A 20-mn tutorial detailed the functioning of the autopilot system logic (user interface, auditory and visual alerts, main flight parameters). In particular the participants were explained the different nominal and off-nominal behaviours. Each off-nominal event was illustrated with a real flight situation in which a conflicting situation occurred and confused the crew. The participants were then asked to comment each slide of the tutorial and to detail precisely all the automation behaviours, the associated knobs and information displayed in the cockpit. When the participants succeeded to recall at least twice the autopilot logic and the user interface, they sat in the flight simulator (left seat) and completed an 1-hour training. They were first trained to perform basic flight maneuvers such as takeoffs and landings and were instructed about the different maximum speeds of the aircraft (flaps 2/takeoff maximum speed =180 knots; flaps 1 = 220 knots, flaps 3 = 320 knots). The training then focused on the interaction with automation: different exercises were performed such as trajectory programing following the ATC instructions: the participants were instructed to repeat the ATC clearances after programing the Flight Control Unit, as it is the case in real flight conditions (e.g. ATC: "Supaero32, steer 320°, climb 3000 ft", participant: "steering 320°, and climbing 3000 ft, Supaero32"). Eventually the three possible critical situations (inconsistent programing, autopilot automatic disconnection, near overspeed mode reversion) were provoked and the participants had to comment, to explain the automation logic and to recover from the situations. At the end of the training, when the participants were successfully managing the autopilot system, they were asked to repeat the autopilot logic, the functioning of the Flight Control Unit and the different auditory warnings. The 15-mn experimental scenario was then started.

4.2.6 Measurements

The flight scenario was recorded *via* a video camera mounted in the cockpit for post hoc analysis. After the end of the scenario, each participant was debriefed with a questionnaire. The questions were:

1. What happened just after takeoff when the ATC required to perform a level-off and what did you do?
2. What happened at 11000 ft during the altitude capture and what did you do?
3. What happened during the descent and what did you do?

We recorded flight data as flight and vertical speeds, mode selections, actions on knobs and buttons. The analysis of these data, together with the observations of the experimeters, allowed us to characterize conflict occurrences, conflict detections and conflict solvings. For instance a conflict situation was characterized by the fact that the participant had failed to manage the vertical separation (the minimum vertical separation between two aircraft is 500 ft).

4.2.7 Experimental data processing

The experimental data were processed so as to characterize the results through four parameters (see next tables): conflict occurrence, conflict detection, time to first relevant action, conflict solving. Conflict occurrence (conflict: Yes) represents the fact that the participant actually faced a situation that was a *priori* identified as critical by the formal analysis: the relevant flight parameters are processed so as to detect the firing of one of the three "hidden" transitions (T1, T2 or T3). Conflict detection (conflict detection: Yes) was evaluated thanks to the participant's real time reactions showing they were aware of the conflict, i.e. a statement (e.g. "What is going on here?") or an action that was judged as relevant for conflict solving by the experimenters - in that case the time to the occurrence of the first relevant action is also provided. Conflict solving (conflict solved: Yes) was evaluated thanks to objective criteria that were *a priori* chosen as conflict markers (e.g. overshooting a 500 ft-segregation). Those criteria were specially defined for each specific conflicting situation. If there is no conflict occurrence at all, the experience is useless for the hypothesis validation (see section 4.2.1). If, for a conflicting situation that actually occurs, some participants fail to solve the conflict the hypothesis is validated. Therefore the conflict detection and the time to the first relevant action are ancillary data that are useful for the characterization of the conflict dynamics, but they are not used for supporting the formal analysis.

4.2.8 Results

The results of the experiments are shown in Tables 1, 2 and 3 for the three situations.

Near overspeed mode reversion (T1)

The participants were asked by the ATC to level off because of some incoming traffic at 600 ft above them. We detected the occurrence of the conflict when the autopilot mode changed from "Level Off" to "Climb +1000 ft/mn (T1). The conflict was solved if the participants managed to respect the 500 ft-segregation. The conflict remained unsolved if they flew at more than 100 ft above the levelling off altitude (see figure 4.3).
Only one participant (participant 5, see table 4.1) anticipated the situation and managed to avoid the conflict as he decided to reduce speed: indeed he changed the aerodynamic configuration of the aircraft, and no "near to overspeed" event was triggered. All the other participants faced a conflict and only one of them solved it.

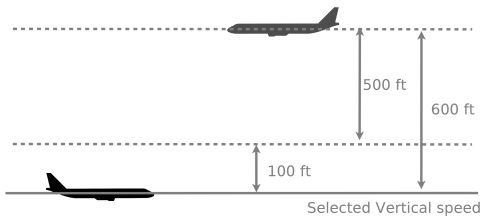


Figure 4.3: Conflict solved criterion for T1.

Participant	Conflict	Conflict detection	Time to first relevant action (s)	Conflict solved
1	Yes	No	–	No
2	Yes	No	–	No
3	Yes	No	–	No
4	Yes	No	–	No
5	No	–	–	–
6	Yes	Yes	–	No
7	Yes	No	–	No
8	Yes	Yes	9	Yes
9	Yes	No	–	No
10	Yes	No	–	No

Table 4.1: Near overspeed mode reversion (T1).

Autopilot automatic disconnection (T2)

We detected the occurrence of the conflict when the autopilot was disconnected by the overspeed event (T2). The conflict was solved if the participants managed to respect the 500 ft-segregation. The conflict remained unsolved if they flew more than 500ft above or under the levelling off altitude (see figure 4.4).
All the participants detected the conflict (see table 4.2) and seven of them solved it.

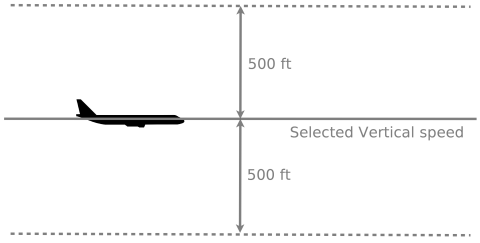


Figure 4.4: Conflict solved criterion for T2.

Participant	Conflict	Conflict detection	Time to first relevant action (s)	Conflict solved
1	Yes	Yes	9	No
2	Yes	Yes	6	No
3	Yes	Yes	26	Yes
4	Yes	Yes	20	Yes
5	No	Yes	18	Yes
6	Yes	Yes	3	Yes
7	Yes	Yes	25	Yes
8	Yes	Yes	10	Yes
9	Yes	Yes	37	No
10	Yes	Yes	7	Yes

Table 4.2: Autopilot automatic disconnection (T2).

Near overspeed mode reversion with inconsistent programming (T3)

The participants were requested to perform a descent. We detected the occurrence of the conflict when the autopilot mode changed from "Descent -1000 ft/mn" to "Level Off" (T3). In this case, the conflict remained unsolved if the participant flew more than 500ft above or under the desired descent trajectory (see figure 4.5). Subject 10 did not correctly execute the ATC speed instruction and never flew "near overspeed" (see table 4.3). Eight participants out of nine detected the conflict and one of them solved it. Two of them started relevant actions but did not manage to solve the conflict. The other participants did not manage to find a relevant action.

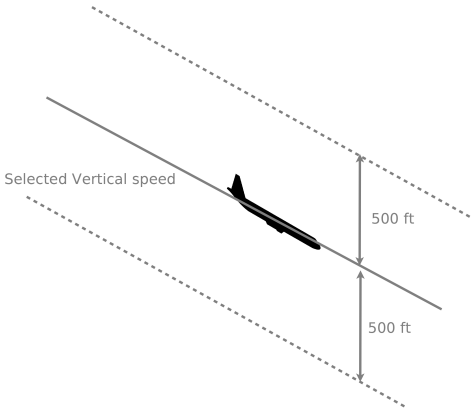


Figure 4.5: Conflict solved criterion for T3.

Participant	Conflict	Conflict detection	Time to first relevant action (s)	Conflict solved
1	Yes	Yes	–	No
2	Yes	Yes	–	No
3	Yes	No	27	No
4	Yes	Yes	–	No
5	Yes	Yes	25	Yes
6	Yes	Yes	40	No
7	Yes	Yes	–	No
8	Yes	Yes	–	No
9	Yes	Yes	–	No
10	No	–	–	–

Table 4.3: Near overspeed mode reversion with inconsistent programing (T3).

4.3 Discussion

In chapter 3 we have proposed generic pattern to predict conflicts in Petri net models of human-automation interactions stemming from automated transitions that may not be perceived by the human operator. This prediction is based on the detection of deadlocks and specific patterns in the Petri net model of the interaction. To test this method, we designed an autopilot system that we formally analysed to identify conflicting situations. Three situations were identified and were integrated in a scenario conducted in our flight simulator with ten general aviation pilots. The results of the experiment tend to show that the transitions that we *a priori* identified as critical with the formal analysis indeed generated conflicts in several cases. Moreover this also demonstrates the interest of conducting experiments as the three situations induced different behaviours.

Indeed, in the first conflicting situation 8 out of 9 pilots objectively exceeded the level-off altitude and during the debriefing all of them reported that they had noticed neither the mode reversion nor the level bust. They declared that their workload was high as the aircraft was still in the initial climb phase and that they were particularly focused on programing the FCU to enter the successive ATC clearances. This result is akin to a previous study conducted with the French safety board that had revealed that the interaction with the FCU during critical flight phases consumes attentional resources to the detriment of the supervision of primary flight parameters [RAC+12].

In the second situation, all the participants detected the conflict and only 3 of them did not take the appropriate corrective actions to avoid a level bust. During the debriefing, all of them said that they had particularly focused on the altimeter as the aircraft was very close to the target altitude. For that reason, they were more likely to face the situation as they had noticed that the altimeter continued to increase.

Though this conflict was detected and solved by most of the pilots it remains critical. Indeed in aeronautics, an event considered catastrophic must have a probability of occurrence that is less than 10-9 per flight hour. Moreover, it took more than 18s for half of the pilots to solve the conflict and to stabilize the aircraft whereas the corrective action was very simple (i.e. push the autopilot button). This conflict occurred during a low-workload situation but one has to consider that the participants might have behaved differently in a more complex situation (e.g. a situation with failures). This result also shows that each conflicting situation should be tested under different experimental conditions (e.g. different levels of workload) and this is a clear limitation of the present study, as we did not counterbalance the order of occurrence of the three conflicts.
Eventually the last situation led to typical automation surprise [SMW07] as all the participants detected an unexpected automation behaviour but only one declared that he had understood the situation. All the other participants stated that the autopilot had had a failure. From a formal point of view this conflict combined the firing of two "hidden" transitions, which probably led to a more complex situation for the participants. Consequently the results of the experiment tend to support the formal approach for conflict prediction but also show that conflicts that are identified without distinction by the formal analysis may indeed be different. Therefore experimental analyses of such situations remain necessary in order to correctly assess their criticality and understand their dynamics. Though the results of the experiment are encouraging, one should notice that the sample participants consisted in general aviation (light aircraft) pilots that are not as experienced as transportation pilots to deal with automation.

4.4 Conclusion

The present study shows that another step has to be considered to mitigate the occurrence of human-automation conflicts. Indeed, if the experiments confirm that an automated mode transition effectively leads to a critical situation. The system designer may consider three options. The first one is to propose a new design of the mode transition logic. Though this approach seems to be most efficient, it may not be applicable: for instance, an overspeed event leads to automatically disconnect every existing autopilot systems (see T2-conflict) as no flight control law can fly an aircraft out of its flight envelope. Therefore an alternative option is to improve the H/M interface in terms of feedback and alerting. For instance, it is possible to display an explanation of the conflict to help the crew to understand the automation behaviour. This could be done automatically through the analysis of the events that

have triggered the state transitions and led to the conflicting situation. Such a solution could be particularly relevant for T3-conflict, e.g. “Level-off because speed is excessive and vertical speed is inconsistent”. Eventually a third and complementary approach is to consider recurrent training to instruct the human operators in operational conditions.

It is worth noticing that the proposed formal approach only detects conflicting situations that may remain unsolved because of a lack of appropriate feedback or because of a gap in the human attention. We have no guarantee on the completeness, i.e. whether the formal model detects all the possible conflicts. The causal relations we assume between automated changes, automated hidden or unseen changes, automated hidden or unseen blocking transitions and conflicts is illustrated in figure 4.6.

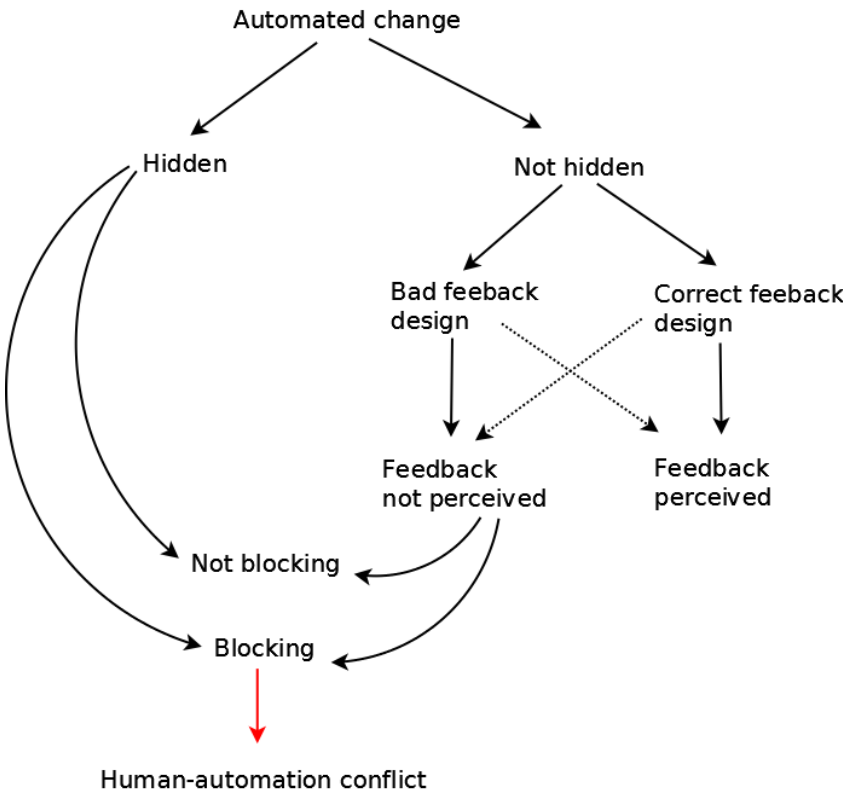


Figure 4.6: Assumed causal relations.

Despite this completeness limitation, the approach appears to be useful in the

frame of the evaluation of the design of the human-machine interaction: even if the human behaviour obviously cannot be predicted in real time, potentially critical situations are identified *a priori*. Nevertheless, detecting conflicts in real time is another challenging problem: indeed in the case of an automated transition with adequate feedback to the pilot there is no certainty about the effect of the feedback on the pilot's situation awareness: as suggested by our results, a conflict may arise and remain undetected or misunderstood. A possibilistic approach [DP90] to model this uncertainty is proposed in chapter 5.

Chapter 5

An estimate of the human assessment of the internal state

5.1 Introduction

In the human-machine interactions studies the problem of the correct human assessment of the situation has been widely discussed. Many different models have been proposed: mental models and situation awareness [End95, RCP99], formal rules of inference [O'B95] or probabilities [OC07]. In this chapter we focus on the human assessment of the *internal state*.

The *observable part of the internal state* (OIS) [OW90] is the part of the internal state on which the human, thanks to the feedbacks, has in principle a perfect knowledge at points in time separated by finite numbers of state changes. In this chapter we provide *an event-based estimate of the human assessment of the observable part of the internal state* (HA-IS). The events we take into account for the HA-IS estimation are the feedbacks and the selections. If no assessment error arises the HA-IS coincides with the OIS. Actually the sending of feedbacks does not guarantee the correct reception of the information, in particular in some cases as the *automated state changes* [Fea05]. Therefore we propose an HA-IS estimation that takes into account the uncertainty related to the two possible situation assessment errors:

- the possible loss of the feedbacks
- the possible wrong human assessment of the initial OIS

In case of incoherences between the HA-IS and the actual OIS, the HA-IS es-

timate could be used to provide a diagnosis of the situation[1].

5.2 A possibilistic model of the HA-IS

Events have effects on the HA-IS, and they can be either *nominal* or *non-nominal*.
Effects are said nominal if they describe the real effect of an event on the internal state (i.e. they describe the actual machine reactions to the events accurately).
Effects are said non-nominal if they describe a possible wrong human assessment of an event effect (i.e. they describe an error model for the human assessment of the internal state, those errors being reactions to events not corresponding to the actual machine behaviour).
In this work we define general rules for the automated characterization of some non-nominal effects. Nevertheless the developed model could allow the specific definition of some non-nominal effects by hand (i.e. characterizing well known assessment human errors).
After the characterization of all the assessment human errors by *non-nominal* effects we see that some events have only a nominal effect (e.g. the human execution of nominal selections) and others (e.g. the feedback sending) may have a nominal effect and non-nominal effects (multi-effect events). The actual value of the OIS can be deduced applying only nominal effects.
We call *HA-IS instance* (or simply *state instance*) a sequence of event effects that is considered as possible. Each time a multi-effect event is fired, several *state instances* are created, one for each effect (see figure 5.1).
We will consider that nominal effects are more plausible than the associated non nominal effects, which allows us to order the plausibilities of the state instances[2].
After the firing of each event it is possible to evaluate the state instance that is the most plausible. Events affect all the state instances, which subsequently evolve in time. Because several multi-effect events may be fired the number of state instances may increase significantly.

The events we take into account are:

- the execution of selections considered as normal
- the execution of a slip[3]

[1] This diagnosis could be useful to compose a proper cognitive countermeasure [DCT11]: a feedback (e.g. a change in the H/M interface) meant to correct human assessment of the OIS.

[2] More details about the chosen plausibility measure are given in section 5.2.1. More details about the assumptions on the effects plausibility are given in section 5.2.2.

[3] In the frame of this study we define a *slip* as the unmeant execution of a selection.

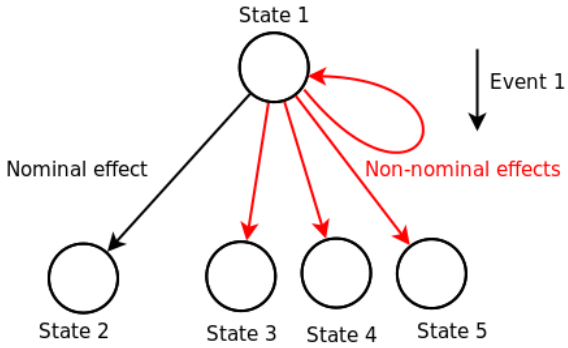


Figure 5.1: Nominal effect and non-nominal effects of Event 1 on the internal state.

- an *automated behaviour*[4] with two possible effects: reception of the relevant feedback (nominal) and loss of the feedback (non-nominal)
- the state initialization with two possible effects: correct initialization (nominal) and wrong initialization (non-nominal)

Initially the most plausible state instance is the one that includes only nominal effects. The effects of this state instance lead to the actual OIS state. We call this particular state instance the *objective state instance*. After the firing of an event considered as unusual in the actual situation the objective state instance is no longer considered as normal. We call this situation an *exception* and the event that led to the exception a *triggering event*. Typical triggering events are selections considered as erroneous in the particular context. If an exception is detected the model determines if there is a non-nominal effect in the past history that, if considered as the actual one (and so the most plausible), could lead to a situation in which the firing of this event is not unusual, but instead normal. We call *exception explanation* this non-nominal effect. If an exception explanation is found its plausibility is considered as normal (instead of its former value). The state instance embedding the exception explanation is considered as the new most plausible one as well. The formerly most plausible state instance, i.e. the *objective state instance*, is no longer coherent with the actual state of affairs (therefore its plausibility is decreased). If an exception explanation is not found the plausibility remains unchanged.

For instance let us consider the case of an automated behaviour changing the internal state from an initial state to a final state: at first the correct reception of the

[4] As seen in section 2.4.2 an *automated behaviour* is an internal state change, with relevant change on the H/M interface, not fired by a selection. We refer to events that trigger those behaviours as "other events" (in opposition to *selections*) or shortly *events*.

relevant feedback is considered as the most plausible effect. If later on the human performs a selection that is a slip (and so producing an exception) in the actual new state, but that in the former state is totally normal, our model identifies the exception explanation in the loss of the relevant feedback (see figure 5.2).

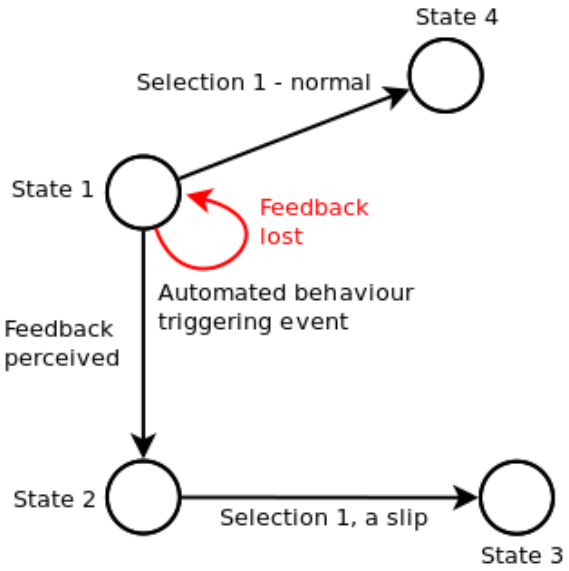


Figure 5.2: Loss of feedback as an exception explanation.

Note that an exception for which an exception explanation has been found is considered by our model as a *possible knowledge conflict* generated because of the exception explanation, and so it will be labelled as a *conflict*. If no explanation is found for the exception it will be labelled just as an *exception*.

The model we propose is based on possibility theory [DP90, DP09] and is represented *via* enhanced logical tables (see section 7.3.1). More details are given in the following sections.

5.2.1 Possibility theory

Possibility theory [DP07][5] is an uncertainty theory devoted to the handling of incomplete information. As such, it complements probability theory in so far as it

[5]The following paragraph is largely taken from the reference.

can capture partial ignorance. Besides, it is not additive and makes sense on ordinal structures.
A possibility distribution is a mapping π from a set of states of affairs S to a totally ordered scale such as the unit interval $[0, 1]$. The function π represents the knowledge of an agent (about the actual state of affairs) distinguishing what is plausible from what is less plausible, what is the normal course of things from what is not, what is surprising from what is expected. It represents a flexible restriction on what the actual state of affairs is, with the following conventions:

- $\pi(s) = 0$ means that state *s* is rejected as impossible
- $\pi(s) = 1$ means that state *s* is totally possible (= plausible, unsurprising, normal)

If the state space is exhaustive, at least one of its elements should be the actual world, so that at least one state is totally possible. If at some point there is no state that is totally possible a re-normalization of the possibility values should be performed. Distinct values may simultaneously have a degree of possibility equal to 1. Possibility theory is driven by the principle of minimal specificity. It states that any hypothesis not known to be impossible cannot be ruled out.

Qualitative possibility relations can be partially specified by a set of constraints of the form $\pi(e_a) > \pi(e_b)$ ("the possibility of e_a is greater than the possibility of e_b"), where e_a and e_b are events. A plausibility ordering on S can be obtained by assigning to each state of affairs its highest possibility level in agreement with the constraints [BDP97].

Qualitative possibility relations can be represented by (and only by) possibility measures ranging on any totally ordered set (especially a finite one [Dub86]).

5.2.2 The assumptions for the model definition

Starting from the definition of the event effects, and from a possibility ordering of those effects, we design a qualitative possibilistic model that can derive all the state instances and their possibility degrees. We build our set of constraints (needed for the qualitative approach) on assumptions concerning the structure of the model, the event effects possibility and the state instances possibility. We have already presented some of those assumptions in section 5.2, the complete list is:

Structure of the model

- the human knowledge of the machine behaviour is correct

This assumption implies that the human is sufficiently trained to use the machine and knows its behaviour. This assumption may be later relaxed so as to enrich the model with the most common human errors.

Event effects possibility

1. the human can possibly loose feedbacks
2. human selections with no effect on the OIS are considered as slips
3. the loss a feedback is more likely to happen than a slip or a mistake[6]
4. the human's knowledge of the initial state is uncertain, but is likely to coincide with the real one

Regarding the first assumption, two effects are taken into account: the feedback is correctly received or the feedback is lost. The expert knowledge about the most common human misinterpretations of the feedbacks may enrich the model, relaxing this assumption.
The second assumption implies that the human does not execute useless selections on purpose.
The observation of the results of a real experience (see section 5.4) persuaded us to include the third and the fourth assumptions.
In our model, thanks to the first assumption, every time a feedback is sent we compute two different instances of the HA-IS estimate: the most possible one is the correct reception of the feedback, the other instance (feedback lost) does not evolve from the previous state. So the uncertainty about the HA-IS estimation increases every time a feedback is sent.

State instances possibility degrees

1. the loss of n feedbacks is more likely to happen than the loss of $n + 1$ feedbacks, for n bounded
2. the loss of n feedbacks is more likely to happen than slips, for n bounded

[6]Whereas a slip is the execution of an unmeant selection, a mistake is a state change (not necessary selection driven) leading to a state defined as erroneous by the system designer.

The first assumption is meant to guarantee different state instances with a different number of lost feedbacks to have a different possibility value: the instance with the least number of lost feedbacks has to be held as the most possible.
The second is meant to compare state instances with multiple lost feedbacks with state instances with slips.
The constraint on n to be bounded on both assumptions is meant to have a finite event set, which is a condition for the qualitative representation [Dub86].

Thanks to the assumptions on the possibility ordering we can explicitly specify a set of constraints on the event effects of the form $\pi(e_a) > \pi(e_b)$, where e_a and e_b are elementary events:

$$\pi(e_f) = \pi(e_{normal}) = 1 > \pi(e_l)$$

$$\pi(e_l) > \pi(e_s) = \pi(e_{unusual}) \doteq \epsilon$$

$$\pi(e_{0c}) = \pi(e_{normal}) > \pi(e_{0w}) = \pi(e_{unusual})$$

Where we denote by:

$e_f \doteq$ the correct reception of a feedback

$e_l \doteq$ a lost feedback

$e_s \doteq$ the arising of a slip or a mistake

$e_{0c} \doteq$ the correct initialization

$e_{0w} \doteq$ a wrong initialization

$e_{normal} \doteq$ a generic event considered as normal

$e_{unusual} \doteq$ a generic event considered as unusual

Note that nominal event effects and normal events must not be confused (see figure 5.3). The words *Normal, Unusual* are used to define the plausibility. The words *Nominal, Non-nominal* are used to define the actual machine model and the error model. Event effects are said *nominal* if they may affect the OIS, and *non-nominal* if they model assessment human errors. Events are defined as *normal (or unusual)* by the designer of the model. As a general rule *non-nominal* effects are considered less plausible than the associated nominal effects (so they are *less than normal*).[7]

[7] In this model only the normal events are modelled *via* multi-effect events, but the same can be done with events with a plausibly that is less than normal.

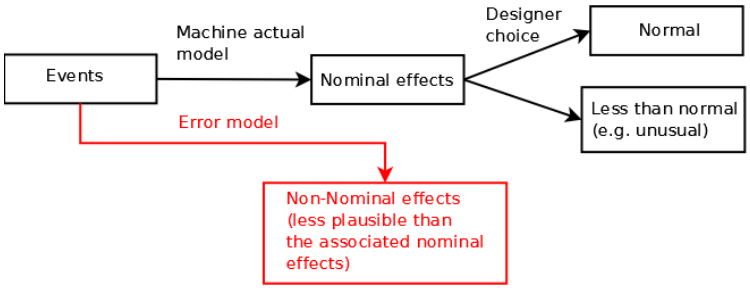


Figure 5.3: Nominal events and non nominal events (and associated plausibilities).

Thanks to the assumptions on the possibility ordering for effects and state instances we can explicitly specify a set of constraints on the state instances of the form $\pi(A) > \pi(B)$, where A and B are state instances:

$\pi({e_l}^n) > \pi(e_s) = \pi(e_{unusual}),\ n < n_{max}$

$\pi({e_l}^n) > \pi({e_l}^{n+1}),\ n < n_{max}$

Where we denote by:

e_l^n = state instance composed by the effect "feedback lost" with multiplicity n

Combining the relations:

$$\begin{aligned} 1 = \pi(e_{normal}) > \pi(e_l) > \pi({e_l}^2) > ... \\ \pi({e_l}^{n_{max}-1}) > \pi(e_{unusual}) = \pi({e_l}^{n_{max}}) > 0 \end{aligned} \tag{5.1}$$

Note that we represent the state instances with braces (as a set) when assessing their possibility, and with parentheses when the order of the effects matters for the resulting final HA-IS state value evaluation. For the effects of a state instance (after the firing of m events) on the HA-IS we note:

$i_m = (e_1, e_2, e_3, ...e_m) \doteq$ state instance

$X \doteq$ internal state

$X(i_m) \doteq$ the value of the internal state resulting from the state instance

For the purpose of the possibility degree evaluation a state instance after the firing of m effects is expressed as a multiset of effects (the multiplicities of the events are denoted by μ with the same subscript of the relevant event):

$$i_m = \{e_1, e_2, e_3, ...e_m\} = \{e_{0c}^{\mu_{0c}}, e_{0w}^{\mu_{0w}}, e_s^{\mu_s}, e_l^{\mu_l}, e_f^{\mu_f}\}$$

To evaluate the possibility of a state instance we use the rules[8]:

$$\text{if } (\mu_{0c} = 1) \wedge (\mu_s = 0) \wedge (\mu_l = 0) \Rightarrow \pi(i_m) = \pi(e_{normal}) = 1$$

$$\text{if } (\mu_{0c} = 1) \wedge (\mu_s = 0) \wedge (0 < \mu_l < n_{max}) \Rightarrow \pi(i_m) = \pi({e_l}^{\mu_l}) \doteq \lambda^{\mu_l}$$

$$\text{if } (\mu_{0w} = 1) \vee (\mu_s = 1) \vee (\mu_l \geq n_{max}) \Rightarrow \pi(i_m) = \pi(e_{unusual}) = \epsilon$$

For a sequence of m events there are several sequences of m effects, i.e. several state instances. Hereafter different state instances are marked by the index r, so ${i_m}^r$ represents the r-th state instance composed of m effects. The normalization rule used is (hereafter we note the normalized π shortly as π^{norm}):

$$\max_r(\pi({i_m}^r)) < 1 \wedge r_M = \arg\max_r(\pi({i_m}^r)) \Rightarrow \pi({i_m}^{r_M}) \leftarrow 1$$

This normalization is meant to assign the *normal possibility degree* (i.e. the maximum value, 1) to the state instance with the higher possibility degree after the firing of m events (it may have a possibility degree smaller than the maximum), whose index is r_M.

It may be interesting to compute the *possibility for the HA-IS to assume a value*, defined as:

$$\Pi(X = value_j, m) \doteq \max_r(\pi({i_m}^r)) : X({i_m}^r) = value_j$$

With this equation we define the possibility degree of a set defined as *the set including all the state instances bringing to state X after the firing of* m *events*: the possibility degree of this set is defined as *the maximum possibility degree of all the state instances belonging to the set*.

We finally denote by:

${i_m}^{obj} \doteq$ objective state instance after the firing of m events

$HA - {IS_m}^r \doteq X(i_m^r)$, we define each of the possible HA-IS estimates as the *internal state* resulting in the application of a possible state instance (i.e. the internal state resulting of this possible sequence of events)

$OIS_m \doteq X({i_m}^{obj})$, because the OIS corresponds to the HA-IS estimate resulting in the application of the objective state instance

In the following sections a mock-up example and real case example are detailed.

[8]Those rules could be defined also as what is called a *leximin selection for the multiset, based on the multiplicity of* e_l. Leximin is a function similar to the minimum that may discriminate sets whose minimum is the same. The idea is to compare two sets at first via the simple minimum function, and if the sets have the same minimum, count the multiplicity of the minimal values and choosing as lexi-minimal the set with greater multiplicity. For a detailed definition of leximin see [DF05].

5.3 A three-state machine

This simple example is meant to show the effects of selections and automated behaviours on the HA-IS estimate. Let us consider a machine with only one internal state variable with three values *high, medium, low*, two selections *up, down* and four automated behaviours *high to medium, medium to low, low to medium, medium to high*. For the model of this example see table 5.1: a logic table enhanced by three additional rows:

- the *Nominal* row takes value 1 if the column is a nominal mode, 0 if it is a non nominal mode
- the *Non nominal version of* row specifies, for non nominal modes, which is the column describing the nominal version of this mode
- the *Possibility* row specify the possibility value for this column

5.3.1 The effect of selections on the HA-IS

In this scenario we show the effect of selections on the HA-IS estimate: starting from the *medium* value two *up* selections will be performed in sequence.

Therefore the initial value for the OIS is:

$$Variable = medium$$

For HA-IS we have three initial instances.
First instance (normal):

$$\pi({i_1}^1) = 1$$
$$Variable = medium$$
$${i_1}^1 = \{e_{0c}\}$$

Second instance (wrong initialization):

$$\pi({i_1}^2) = \epsilon$$
$$Variable = high$$
$${i_1}^2 = \{e_{0w}\}$$

Third instance (wrong initialization):

$$\pi({i_1}^3) = \epsilon$$
$$Variable = low$$

		c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14
SITUATION															
Selection	Up	1	1	1											
	Down				1	1	1								
Event	High to Medium							1				1			
	Medium to Low								1				1		
	Low to Medium									1				1	
	Medium to High										1				1
State															
Variable	Low	1			1					1					
	Medium		1			1			1		1				
	High			1			1	1							
BEHAVIOUR															
State															
Variable	Low				1	1			1						
	Medium	1					1	1		1					
	High		1	1							1				
POSSIBILISTIC															
	Nominal	1	1	1	1	1	1	1	1	1	1	0	0	0	0
	Non nominal version of											c7	c8	c9	c10
	Possibility	normal	normal	slip	slip	normal	normal	normal	normal	normal	normal	lost feedback	lost feedback	lost feedback	lost feedback
	Description	Selection Up, from Low to Medium	Selection Up, from Medium to High	Selection Up, no effect because already High	Selection Down, no effect because already Low	Selection Down, from Medium to Low	Selection Down, from High to Medium	Automated behaviour High to Medium	Automated behaviour Medium to Low	Automated behaviour Low to Medium	Automated behaviour Medium to High	Automated behaviour High to Medium, but unseen	Automated behaviour Medium to Low, but unseen	Automated behaviour Low to Medium, but unseen	Automated behaviour Medium to High, but unseen

Table 5.1: Enhanced logic table for the three-state machine.

$i_1{}^3 = \{e_{0w}\}$

The resulting initial possibility for the HA-IS to assume each of the values is:

$\Pi(Variable = high) = \epsilon$
$\Pi(Variable = medium) = 1$
$\Pi(Variable = low) = \epsilon$

After the execution of an *up* selection the OIS is:

$Variable = high$

For the first HA-IS instance (the normal one) the execution of an *up* selection has only a nominal effect that is considered as a normal selection execution. The first HA-IS instance becomes:

$\pi(i_2{}^1) = 1$
$Variable = high$
$i_2{}^1 = \{e_{0c}, e_n\}$

For the second HA-IS instance the execution of an *up* selection has only a nominal effect that is considered as a slip[9]. The second HA-IS instance becomes:

$\pi(i_2{}^2) = \epsilon$
$Variable = high$
$i_2{}^2 = \{e_{0w}, e_s\}$

For the third HA-IS instance, the execution of an *up* selection has only a nominal effect that is considered as a normal selection execution. The third HA-IS instance becomes:

$\pi(i_2{}^3) = \epsilon$
$Variable = medium$
$i_2{}^3 = \{e_{0w}, e_n\}$

The resulting possibility distribution for HA-IS is:

$\Pi(Variable = high) = 1$
$\Pi(Variable = medium) = \epsilon$

[9]Remember that nominal effects and normal effects must not be confused. Effects are said *nominal* if they may affect the OIS. Moreover if an event in a given situation has only one possible effect, this effect has to be nominal. Effects are defined as *normal* by the designer of the model. For instance in this case the execution of the *up* selection in the state value *Variable-high* has the nominal effect to keep the variable value unchanged, nevertheless this useless selection with no actual outcome is considered has a slip, so not normal.

$\Pi(Variable = low) = 0$

At this point the most possible estimation (normal case) for the HA-IS is *Variable-high*, as for the actual value of the OIS.

After the execution of the second *up* selection the OIS is unchanged:

$Variable = high$

The first instance now includes an unusual event, the slip. Its possibility value is decreased to ϵ:

$\pi({i_3}^1) = \epsilon$
$Variable = high$
${i_3}^1 = \{e_{0c}, e_n, e_s\}$

The second instance:

$\pi({i_3}^2) = \epsilon$
$Variable = high$
${i_3}^2 = \{e_{0c}, e_s, e_s\}$

The third instance:

$\pi({i_3}^3) = \epsilon$
$Variable = high$
${i_3}^3 = \{e_{0c}, e_n, e_n\}$

The resulting possibility distribution for HA-IS is:

$\Pi(Variable = high) = \epsilon$
$\Pi(Variable = medium) = 0$
$\Pi(Variable = low) = 0$

The execution of a second *up* selection has no effect on the internal state, because the maximum value for the variable has already been reached. Then two cases are possible:

- The human knew to have already reached the maximum value and by accident performed another *up* selection, a slip (as in the first and second state instances);
- The human thought that the initial value of the variable was low, they performed a first *up* selection to reach the medium value, and then performed a second *up* selection to reach the high value (as in the third state instance) .

Because there is no state that is totally possible a normalization of the possibility values has to be performed. The resulting possibility distribution for HA-IS is:

$$\Pi(Variable = high) = 1$$
$$\Pi(Variable = medium) = 0$$
$$\Pi(Variable = low) = 0,$$

which is the final possibility for this scenario. A summary of this scenario is given on table 5.2.

	event-1	effect	variable	π	event-2	effect	variable	π	event-3	effect	variable	π	π^{norm}
OIS													
	initialization		medium		selection up		high		selection up		high		
HA-IS													
		e_{0c}	medium	1		e_n	high	1		e_s	high	ϵ	1
		e_{0w}	high	ϵ		e_s	high	ϵ		e_s	high	ϵ	1
		e_{0w}	low	ϵ		e_n	medium	ϵ		e_n	high	ϵ	1
variable													
high				ϵ				1				ϵ	1
medium				1				ϵ				0	0
low				ϵ				0				0	0

Table 5.2: State instances, evolution of the HA-IS and relevant possibility values.

Only the tabular representation will be given for the next two scenarios of this example.

5.3.2 The effect of automated behaviours on the HA-IS

In this scenario we show the effect of automated behaviours on the HA-IS estimate: starting from the *medium* value one *medium to high* event is fired. The evolution of the HA-IS is given in table 5.3.

The firing of event *medium to high* has two effects: the nominal (and normal) effect is to actually update the value of the HA-IS from *medium* to *high*; the non-nominal (and non-normal) effect is to leave the value of the HA-IS unchanged. Our model applies effects only to state instances compatible with their input state. So this particular nominal effect may affect only state instances whose variable value is *medium* because it *requires* the *medium* value as an input state[10].

[10]If that was not the case our model would have taken as possible also an absurd situation in which the human estimates the machine state to be *low* and successively takes note of *the medium to high* state change *without rectifying their prior assessment error*. In fact if they rectify it, the state instance to represent their actual internal state assessment is the nominal one, which already exists.

	event-1	effect	variable	π	event-2	effect	variable	π
OIS								
	initialization		medium		medium to high		high	
HA-IS								
		e_{0c}	medium	1		e_f	high	1
		e_{0c}	medium	1		e_l	medium	λ
		e_{0w}	high	ϵ		e_l	high	ϵ
		e_{0w}	low	ϵ		e_l	low	ϵ
variable								
high				ϵ				1
medium				1				λ
low				ϵ				ϵ

Table 5.3: State instances, evolution of the HA-IS and relevant possibility values.

5.3.3 The effect of a selection on the HA-IS after the execution of an automated behaviour

In this scenario we show the effect of a selection after the execution of an automated behaviour (the effects of this automated behaviour are shown in the previous subsection).

	event-1	effect	variable	π	event-2	effect	variable	π	event-3	effect	variable	π	π^{norm}
OIS													
	initialization		medium		medium to high		high		selection up		high		
HA-IS													
		e_{0c}	medium	1		e_f	high	1		e_s	high	ϵ	ϵ
		e_{0c}	medium	1		e_l	medium	λ		e_n	high	λ	1
		e_{0w}	high	ϵ		e_l	high	ϵ		e_s	high	ϵ	ϵ
		e_{0w}	low	ϵ		e_l	low	ϵ		e_n	medium	ϵ	ϵ
variable													
high				ϵ				1				λ	1
medium				1				λ				ϵ	ϵ
low				ϵ				ϵ				0	0

Table 5.4: State instances, evolution of the HA-IS and relevant possibility vale.

After the execution of the *up* selection the OIS is unchanged. Because there is no state that is totally possible a normalization of the possibility values has to be performed. The resulting possibility distribution for HA-IS shows that the most possible instance is (represented in the second row of the HA-IS table):

- Correct initialization, at that point this state instance is normal (possibility degree $= 1$)
- A lost feedback, at this point the state instance has possibility degree λ

- A nominal and normal selection, so the possibility degree remains λ

Therefore for our model the most possible sequence of events includes a non-nominal effect: because of an automated behaviour (corresponding to the event-2) the feedback sent from the automation has been lost.

5.4 A flight simulator mission

In this section we present a real case application of the model, i.e. the logic of the autopilot used for the experience described in chapter 4. As for the three-state machine example presented in section 5.3, the first step is the definition of the enhanced logic table describing the events and the possible effects. The construction of an enhanced logic table could be easily automatized starting from the logic table and following the rules:

- Identify the automated behaviours. For each of them identify all the changes in the OIS. For each change in the OIS executed by this automated behaviour, create a non nominal effect (described by a new column) to represent the loss of the relevant feedback. For the non-nominal effects record the column corresponding to the nominal version of it in the line *non nominal version of*.
- define as unusual the execution of a slip (i.e. selection with no effect on the OIS)
- define as nominal all the other effects

After the creation of this automatically generated enhanced logic table, the designer of the model could enrich the model defining some more columns as slips (or generically errors). Those columns describe non necessary selections with no consequences, they may describe selections with undesired effects on the OIS, or also undesired automated behaviours.

On the other hand to recover the logic table from the enhanced logic table, just erase the columns corresponding to non-nominal events, and erase the rows *nominal, non nominal version of, possibility*. For the enhanced logic table of the autopilot see table 5.5. Hereafter we detail the state variables of the model, the events and the effects.

State variables

AP state (On/Off)

ATHR state (On/Off)

		c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19	c20	c21	c22	c23	c24
SITUATION																									
Selections	AP button	1	1	1																					
	ATHR button																		1	1					
	Control stick activation												1	1											
	Control stick desactivation														1										
	Throttle lever activation															1	1								
	Throttle lever desactivation																	1							
Event	Speed becomes < Vls										1	1													1
	Speed becomes Norm						1	1																	
	Speed becomes > Vmax-5								1	1															
	Speed becomes > Vmax				1	1																		1	
	Vertical speed divergence > 500 ft																				1	1	1		
STATE																									
Auto Pilot	On			1										1							1	1		1	
	Off		1										1										1		
Autothrust	On																1			1					
	Off															1			1						
Speed	<Vls	1																							
	Norm		1	1																		1			
	>Vmax-5		1	1																	1				
	>Vmax	1																							
Control stick	Active														1										
	Inactive																								
Throttle lever	Active																								
	Inactive																	1							
BEHAVIOUR																									
State																									
Auto Pilot	On		1																						
	Off			1	1						1													1	1
Autothrust	On																		1						
	Off																			1					
Speed	<Vls										1														
	Norm						1																		
	>Vmax-5								1																
	>Vmax				1																				
Control stick	Active												1												
	Inactive														1										
Throttle lever	Active															1									
	Inactive																	1							
POSSIBILISTIC																									
	Nominal	1	1	1	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0
	Non nominal version of					4,6,8,10		4,6,8,10		4,6,8,10		4,6,8,10												4,6,8,10	4,6,8,10
	Possibility	**slip**	normal	normal	normal	**lost feedback**	normal	**lost feedback**	normal	**lost feedback**	normal	**lost feedback**	normal	**slip**	normal	normal	**slip**	normal	normal	normal	**slip**	**lost feedback**	normal	**lost feedback**	**lost feedback**
	Description	AP button On when Speed¿Vmax or Speed<Vls!"	AP button On when Speed=Norm et AP off (→On)	AP button On when Speed=Norm et AP on (→Off)	Exceeding speed	Exceeding speed, but unseen!	Speed becomes Normal	Speed becomes Normal, but unseen!	Speed becomes >Vmax-5	Speed becomes >Vmax-5, but unseen!	Speed becomes ¡Vls	Speed becomes <Vls, but unseen!	Control stick On when AP off	Control stick On when AP on!	Control stick Off	Throttle lever On when ATHR off	Throttle lever On when ATHR on!	Throttle lever Off	ATHR button when ATHR off (→ATHR on)	ATHR button when ATHR on (→ATHR off)	Vertical speed divergence > 200 ft, knowing that AP on and speed>Vmax-5	Vertical speed divergence > 200 ft, unnoticed	Vertical speed divergence > 200 ft but under responsibility of the pilot, because AP off	Exceeding speed, but just AP disconnection perceived!	Speed becomes <Vls, but just AP disconnection perceived!

Table 5.5: Enhanced logic table for the real case example.

Airspeed (Underspeed, Normal, Near overspeed, Overspeed)[11]

Control stick (Actioning, Not actioning)

Throttle lever (Actioning, Not actioning)

Events

AP button $\doteq$ Autopilot engagement/disengagement button pressed, selection

ATHR button $\doteq$ Autothrust engagement/disengagement button pressed, selection

Control Stick On $\doteq$ Control stick activation, selection

Control Stick Off $\doteq$ Control stick deactivation, selection

Throttle lever On $\doteq$ Throttle lever activation, selection

Throttle lever Off $\doteq$Throttle lever deactivation, selection

Initialization $\doteq$ Creation of all the initial state instances, event

Speed Low $\doteq$ Airspeed takes value Underspeed, event

Speed Normal $\doteq$ Airspeed takes value Normal, event

Near overspeed $\doteq$ Airspeed takes value Near overspeed, event

Overspeed $\doteq$ Airspeed takes value Overspeed, event

Trajectory divergence $\doteq$ divergence between pilot selected trajectory and autopilot executed trajectory that is greater than 250 feet, event

Nominal effects for automated behaviours

Airspeed takes value Underspeed (perceived)

Airspeed takes value Normal (perceived)

Airspeed takes value Near overspeed (perceived)

Airspeed takes value Overspeed (perceived)

[11] "Underspeed/Overspeed" means that the airspeed is smaller/greater than minimum/maximum speed. Minimum and maximum speed are calculated by the autopilot and they depend on the flight envelope. "Near overspeed" means that the speed is between the maximum speed and the maximum speed minus five knots. "Normal" means that the speed is between the minimum speed and maximum speed minus five knots.

Trajectory divergence greater than 250 feet when the AP is off (the pilot is in charge of the flight level)

Selection nominal effects

AP/ATHR connection/disconnection in nominal condition

Control stick/Throttle lever activation/deactivation in nominal condition

Slips and mistakes

AP connection during overspeed/underspeed (slip)

Control stick/Throttle lever activation when AP/ATHR On (slip)

Trajectory divergence consciously greater than 250 feet and increasing because of AP on (mistake)[12]

Non-nominal effects for automated behaviours (lost feedbacks)

Airspeed takes value Underspeed (lost feedback)

Airspeed takes value Normal (lost feedback)

Airspeed takes value Near overspeed (lost feedback)

Airspeed takes value Overspeed (lost feedback)

Airspeed takes value Overspeed but just AP disconnection perceived

Airspeed takes value Underspeed but just AP disconnection perceived

Trajectory divergence greater than 250 feet when AP is on (lost feedback)

The data generated during the experiences used in chapter 4 have been pre-processed to generate sequences of events (among which there are selections). Those event sequences (one sequence for each pilot running the experiment in the flight simulator) have been then processed by our model to automatically detect exceptions, i.e. the objective state instance is no longer considered as normal. In those cases the exception is analysed: the exception explanation (if any is found) is specified by the model using the description of the relevant column in the table,

[12]This effect is defined as a mistake by the designer by hand, so its possibility value is not automatically generated starting from the logic table. By that we mean that *the pilot may not voluntarily be aware of the increasing trajectory divergence and that the AP is on (so it is the cause of the divergence) without taking actions, passively accepting that their requests are not executed.*

and the triggering event is specified as well using the description of the relevant column in the table. That in textual form[13]:

if a non nominal effect (i.e. exception explanation) is found it is labelled as a possible knowledge conflict: `Conflict description: 'Triggering event' because 'exception explanation'`

if a non nominal effect (i.e. exception explanation) is not found it is labelled as a simple exception: `Exception description: 'Triggering event'`

Hereafter the analysis performed for two participants is presented.

5.4.1 Example 1

The sequence of events generated from the pre-processing of the data recorded during the experience with participant 4 is shown hereafter. 57 events (among which there are some selections) have been generated:

'Initialization', 'Control Stick Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Control Stick On', 'Speed Normal', 'Control Stick Off', 'Control Stick On', 'ATHR button', 'Control Stick Off', 'SpeedLow', 'Speed Normal', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'AP button', 'Control Stick Off', 'Near overspeed', 'Control Stick On', 'Control Stick Off', 'Speed Normal', 'Near overspeed', 'Overspeed', 'Near overspeed', 'Speed Normal', 'AP button', 'Near overspeed', 'Trajectory divergence', 'AP button', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'AP button', 'Trajectory divergence', 'AP button', 'Control Stick On', 'Speed Normal', 'Near overspeed', 'Control Stick Off'

The initial values for the OIS are:

AP state: Off

ATHR state: On

[13]The message that is automatically generated by our model could later be used to compose specific feedbacks meant to correct the human internal state assessment. By the way the definition of those feedbacks is out of the scope of this work. Note that the real time version of the algorithm is totally feasible: the computing time for a 15-mn mission is lesser than 1 mn.

Airspeed: Underspeed

Control stick: Actioning

Throttle lever: Not actioning

After the firing of the 25th event, 119.5 seconds from the beginning of the experiment, the model detects an exception:

```
Exception description:  'Control stick On when AP on!'
```

After the firing of the 34th event, 772.6 seconds from the beginning of the experiment, the model detects a conflict:

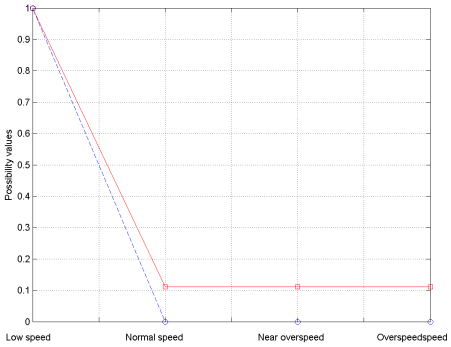
```
Conflict description:  'Vertical speed divergence > 250
    ft, unnoticed' because 'Speed becomes >Vmax-5, but unseen!'
```

After the firing of the 51st event, 886.1 seconds from the beginning of the experiment, the model detects a conflict:

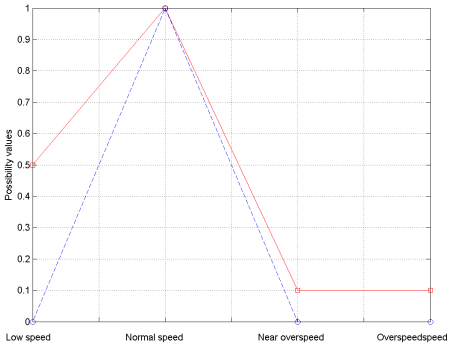
```
Conflict description:  'Vertical speed divergence > 250
    ft, unnoticed' because 'Speed becomes >Vmax-5, but unseen!'
```

After the execution of the 57 events 196 state instances are considered as possible (with different possibility degrees). The computation time is 30 seconds. It may be interesting to show the evolution of the possibility degrees for the HA-IS to assume a value for the state variables. In figures 5.4, 5.5, a summary of the possibility distribution for the airspeed is shown to visualize the shape of a possibility distribution for a typical sequence of events. The possibility distribution is indicated by the red circles. Our possibility evaluation is qualitative, nevertheless in the graphic representation we have arbitrarily assigned quantitative values to the possibilities (in our choice we respect the qualitative ordering) to plot them. The value corresponding to the OIS is indicated by the blue circle with value 1 on the y-axis. Remember that the value of the OIS corresponds to the HA-IS state resulting from the application of the objective state instance. So if no exception arises the most possible estimate (the maximum of the red line) should be the actual state (the blue circle). Moreover the second most possible estimate should be the former value (prior to the last fired event) of the actual state. The third most possible estimate should be the value of the actual state prior to the last two changes and so on.

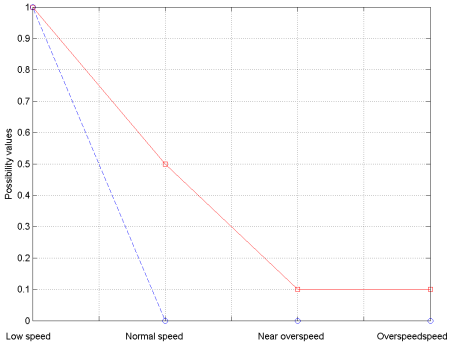
Remember that after the firing of 34 events an exception is detected by the model, and that is graphically highlighted in figure 5.5f: the objective state instance



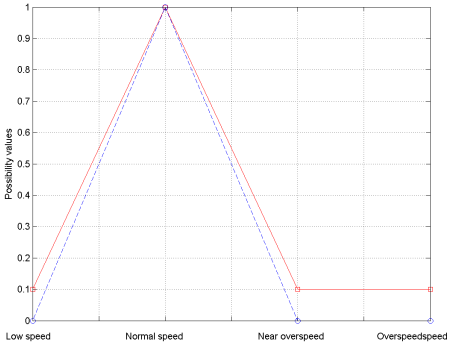
(a) Initial



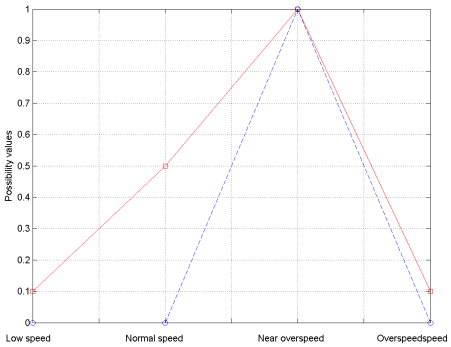
(b) After 12 events



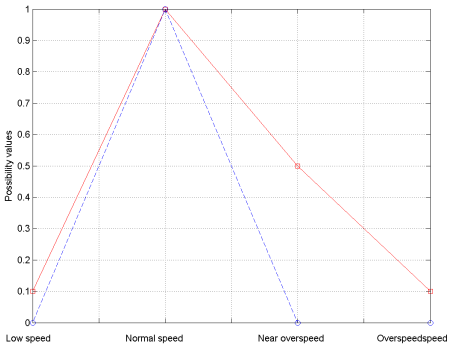
(c) After 17 events



(d) After 22 events

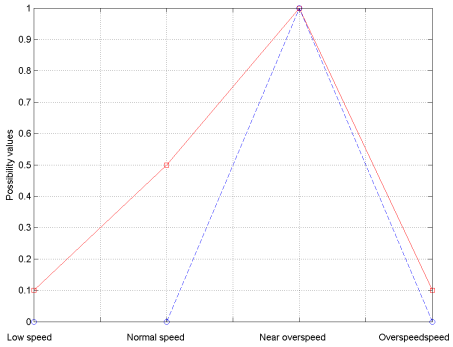


(e) After 24 events

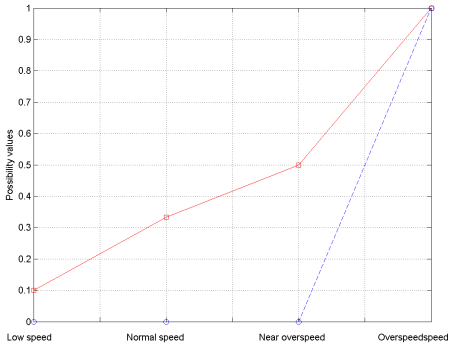


(f) After 27 events

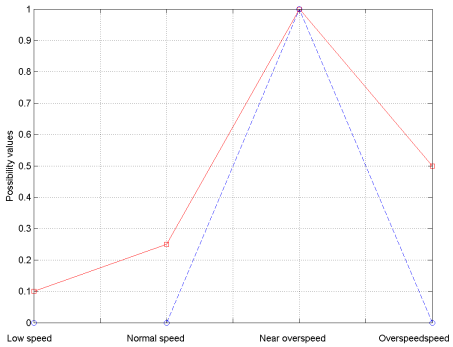
Figure 5.4: Speed possibility distribution



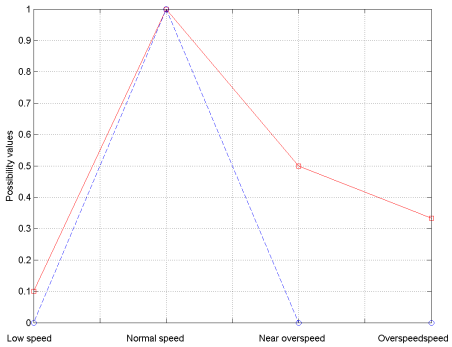
(a) After 28 events



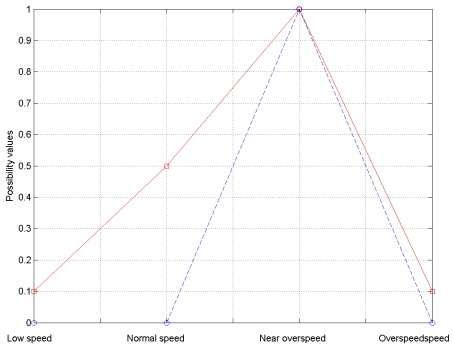
(b) After 29 events



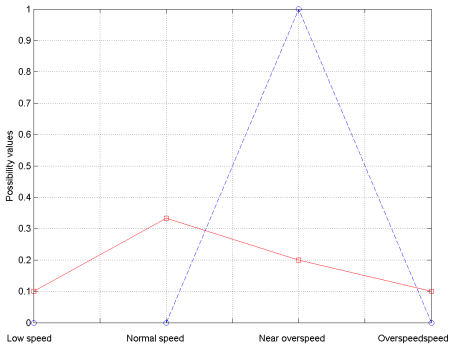
(c) After 30 events



(d) After 31 events



(e) After 33 events



(f) After 34 events

Figure 5.5: Speed possibility distribution

is no longer considered as normal[14]. Also after an exception the most possible estimation for the human assessment of the internal state (which is wrong in this case) is given by the maximum of the red line.

5.4.2 Example 2

The sequence of events generated from the pre-processing of the data recorded during the experience with participant 8 is shown hereafter. 85 events (among which there are some selections) have been generated:

'Initialization', 'Throttle lever On', 'Throttle lever Off', 'ATHR button', 'Control Stick On', 'Throttle lever On', 'Throttle lever Off', 'ATHR button', 'ATHR button', 'Control Stick Off', 'Control Stick On', 'ATHR button', 'Control Stick Off', 'Speed Normal', 'Control Stick On', 'Control Stick Off', 'AP button', 'Near overspeed', 'Speed Normal', 'Near overspeed', 'Overspeed', 'Near overspeed', 'Overspeed', 'Near overspeed', 'Speed Normal', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Control Stick Off', 'Control Stick On', 'Near overspeed', 'Speed Normal', 'Near overspeed', 'Speed Normal', 'Near overspeed', 'Speed Normal', 'Near overspeed', 'Overspeed', 'Near overspeed', 'Control Stick Off', 'Speed Normal', 'AP button', 'Near overspeed', 'ATHR button', 'exceedingVzLevel1', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Control Stick On', 'Speed Normal', 'Control Stick Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'Throttle lever On', 'Throttle lever Off', 'ATHR button'

The initial values for the OIS are:

AP state: Off

ATHR state: On

Airspeed: Underspeed

Control stick: Not actioning

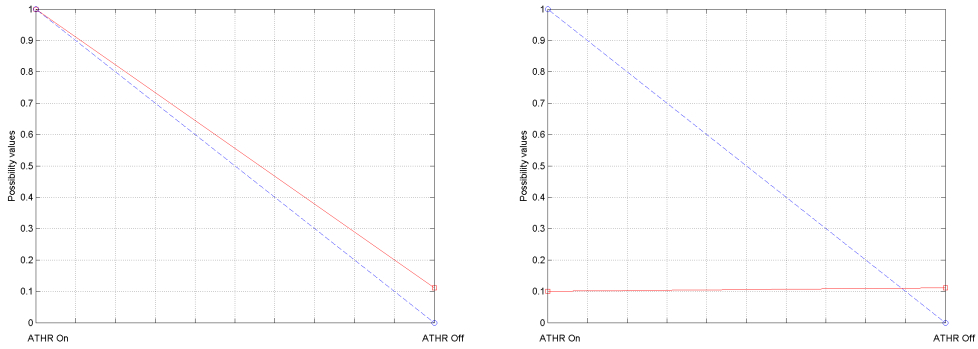
Throttle lever: Not actioning

[14]The possibility for a normal event is 1.

We picked up this second example because of the 2nd event 30.2 seconds from the beginning of the experiment, as the model detects a different exception (see figure 5.6):

```
Conflict description:  'Throttle lever On when ATHR on!'
   because 'Wrong state initialization'
```

Initially the ATHR is on: operating the throttle lever has no effects (this action is considerate as a slip). The model explains this slip as the result of a wrong HA-IS: if the participant's initial assessment of the ATHR state was Off that could explain the execution of this action as nominal. It is worth noticing that after giving up this useless action ('Throttle lever Off') the participant deactivated the ATHR ('ATHR button') and they started again operating the throttle lever ('Throttle lever On'), probably because they had a wrong initial situation assessment and they understood their error.



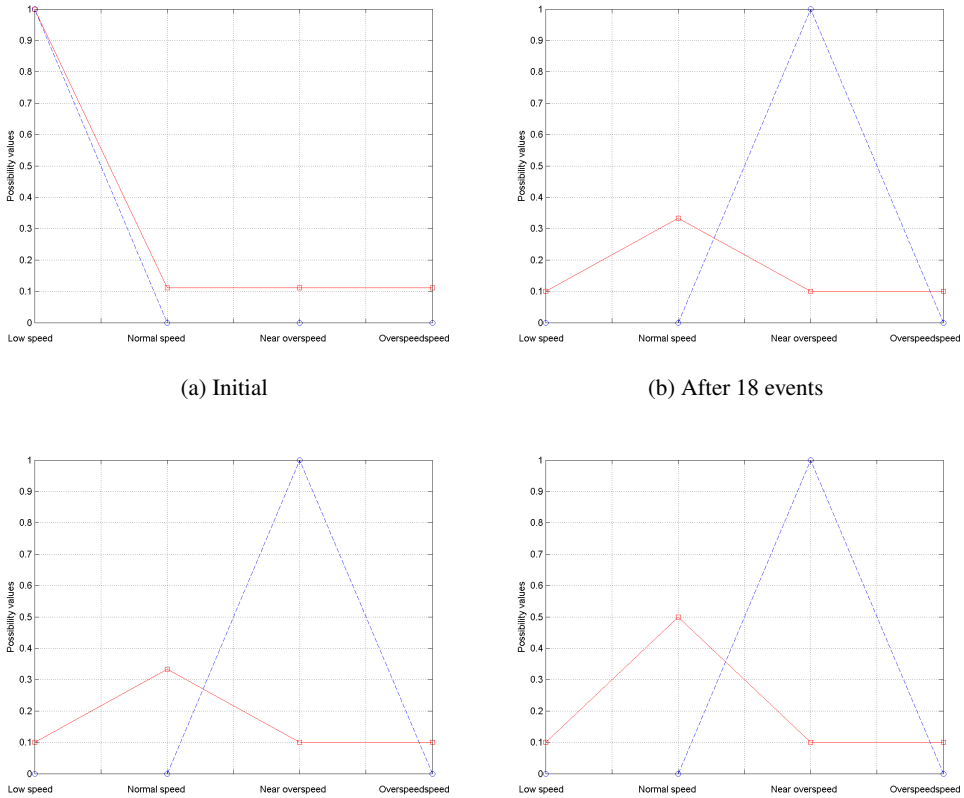
(a) Initial (b) After 2 events

Figure 5.6: ATHR state possibility distribution

The model detects three more instances of the already explained conflict:

```
Conflict description:  'Vertical speed divergence > 250
   ft, unnoticed' because 'Speed becomes >Vmax-5, but unseen!'
```

which is the same conflict identified for participant 4 (see figure 5.7).



(a) Initial

(b) After 18 events

(c) After 46 events

(d) After 49 events

Figure 5.7: Speed possibility distribution

5.5 Conclusion

In this chapter we have presented a possibilistic model that estimates the human assessment of the observable part of the internal state, based on the human selections and on the automated internal state changes. This model can detect exceptions (i.e. unusual sequences of events) and, when possible, give an exception explanation (i.e. a non nominal effect of an event). The *exception detection* could be used as a possible knowledge conflict detection, and the *exception explanation* as conflict identification. This process of detection/identification should be further tested in real time applications in order to verify the validity of the model estimate.

Chapter 6

An attentional tunnelling fuzzy model

6.1 Introduction

The main objective of this work is to develop methods to reduce human-machine conflicting situations *via* prevention. The achievement of this main objective has led us to propose hints for the further development of a real time conflict detection model to be included in a *conflict manager*. A possible general architecture of a human-machine system equipped with a conflict manager is shown in the Introduction, see figure 1.5. In chapter 5 we have proposed an enhanced model with uncertainty management to be used as a real time conflict *detection* and *identification* tool. This uncertainty model could be further enriched thanks to data coming from the observation of the human (thanks to dedicated sensors), and methods to infer the human's "state"[1].
In this chapter we present an approach to characterize attentional tunnelling, which is a relevant human's "state" in aviation [DTCR09a]. More precisely we present a fuzzy model for behavioural and physiological data aggregation based on expert knowledge.

[1]For instance this can be achieved through adapting the plausibilities to the actual estimated human's "state": an inattentive human is more likely to miss feedbacks.

6.2 Attentional tunnelling

A rough definition of attentional tunneling is “the allocation of attention to a particular channel of information, diagnostic hypothesis or task goal, for a duration that is longer than optimal, given the expected cost of neglecting events on other channels, failing to consider other hypotheses, or failing to perform other tasks” [Wic05].
If the human is caught in attentional tunneling they are likely to neglect feedbacks that are useful to correctly assess the internal state, and thus could bring about human-machine conflicts. The relation between attentional tunneling and conflicts is particularly dangerous because conflicts could also be the cause (and not only the consequence) of attentional tunneling (see Introduction).

In the next section we present an experiment that was conducted in order to give further evidence of the relation between conflicts and attentional tunneling. In this experiment data were collected in order to tune and test a fuzzy model to characterize attentional tunnelling for behavioural and physiological data aggregation based on expert knowledge.

6.3 Experimental details

6.3.1 Material

The experimental setup developed at the Institut Supérieur de l’Aéronautique et de l’Espace (ISAE) was composed of a robot equipped with different sensors and a ground station to interact with it. The robot could be operated in “manual” or “supervised” mode. In manual mode, the robot was controlled by the operator with a joystick. In supervised mode, the robot performed waypoint navigation autonomously, but any action of the operator with the joystick let them take over until the joystick was released. The ground station (see figure 6.1) was displayed on a 24-inch screen showing different kinds of information to control and supervise the robot. Note that the operator could not see the robot and only gathered information through the screen.

6.3.2 Experimental scenario

We designed an experimental scenario dedicated to provoke attentional tunneling. The scenario consisted of a target localization and identification task. The target was made of black metal with red stripes superimposed and two short messages written in white on each side (front side “OK”, back side “KO”). The mission

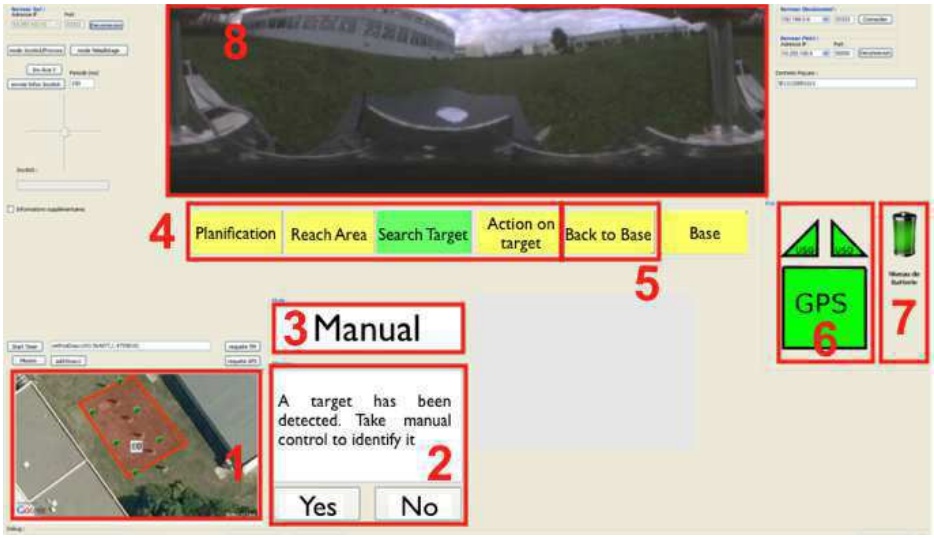


Figure 6.1: Eight Areas Of Interest (AOIs) are defined on the GUI as follows: 1) tactical map, 2) interactive panel, 3) piloting mode, 4) synoptic, 5) back to base, 6) GPS and ultrasound status, 7) battery status, and 8) panoramic video.

lasted around 4 mn and was separated into four main segments: “Reach the area”, “Scan for target”, “Identify target”, and “Battery-Failure”. At the beginning of the mission, the robot navigated in supervised mode to reach the search area. Upon arrival, it started scanning to detect the target. As the robot reached the vicinity of the target, a message was sent to the operator to take over and control the robot in manual mode so as to identify possible similarities in both messages (OK/KO) written on each side of the target. The use of a panoramic video [] and the introduction of a 1-second lag in the control loop increased the task difficulty and favoured excessive focus. While the operator was involved in the identification task, a “low battery event” was sent by the experimenter. This event triggered a safety procedure that made the robot automatically return to base in supervised mode if the operator did not send any command with the joystick. As this failure happened at a crucial moment in the mission when the operator was particularly committed to handling the robot near the target, we expected that the operator would not notice the alerts on the interface warning of the “low battery” event and would persist in achieving the target detection task.

6.3.3 Participants

Twelve healthy adults (2 females, mean age = 28.25, SD = 6.64; mean level of education = 17.41, SD = 2.27), all French defence staff from ISAE who had experience of operating robots, were recruited by local advertisement. All participants gave their informed consent after having received complete information about the nature of the experiment.

6.3.4 Basic behavioural results

The results revealed that 8 participants out of 12 (66.67%) persisted in examining the target instead of letting the robot go back to base. Although they felt surprised by the behaviour of the robot, these participants all declared that they neither noticed the low-battery event nor the other changes on the user interface. The other 4 participants reported that they had noticed the failure and had decided to let the robot go back to base. These subjective results were consistent with the oculomotor measurement that revealed that these participants glanced at the battery icon prior to releasing the joystick.

6.4 Data analysis

According to the literature, an excessive attentional focus of the human is associated with a decreased saccadic activity and long concentrated eye fixations [CBD02] and consequently less scanned areas of interests on the user's interface [TW04].

Attentional tunneling pointer	Reference	Metrics
Allocation of attention to a particular channel of information	[WA09]	Time percentage on Video (PCV)
Decreased saccadic activity and long concentrated eye fixations	[CBD02, TVS+07]	Switching rate (SWR)
Fewer scanned areas of interest on the user's interface	[TW04]	Number of areas of interest (NBAOI)

Table 6.1: Attentional tunneling pointers and relevant metrics.

Because of the relationship between attentional tunneling, stress and increased workload [BFR52, Eas59, WE66, Wil85] other metrics are used to quantify high workload and stress (see table 6.2).

High workload and stress pointer	Reference	Metrics
Heart rate	[CSDP10]	Heart rate (HRF)
Heart rate irregularity	[BR69]	Heart rate irregularity (HRS)

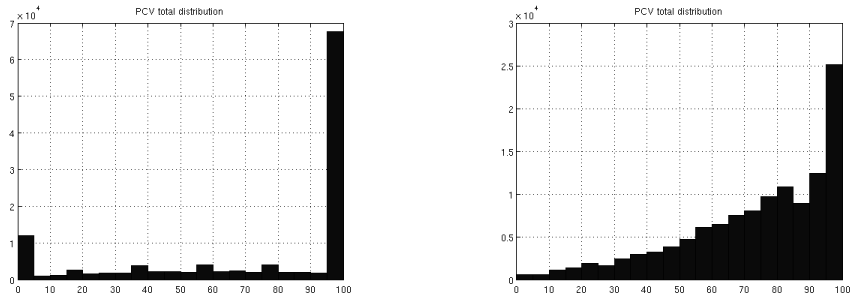
Table 6.2: High workload and stress pointers and relevant metrics.

A description of the metrics we have used is given in the following sections.

6.4.1 Eye tracker data

An eye tracker (ET) is a device to detect the position of the pupil and, thanks to a previous calibration process, the point on the screen the gaze is pointing at. Once a partition of the graphical user interface has been done, the data coming from the ET can be interpreted as a sequence of AOIs (areas of interest) been gazed at at each frame of the sampling. In this work the value of the position as a continuous value is not analysed, neither are all the continuous quantities as the gaze barycenter, speed and acceleration. Hereafter we give a description of the process that brought us to the definition of the three metrics used to analyse the AOI sequence.

Time percentage on Video (PCV) The more the participant spends time gazing at the video (AOI 8), the less the probability they perceive the information from the other AOIs. Of course the dataflow received from the video is richer than the other AOIs. Moreover AOI 8 is the largest. For almost all the participants the most gazed at AOI will be AOI 8. However there are noticeable differences between the participants. In order to compare the time spent on each AOI we could have made a "normalization" taking as the denominator an index taking into account the surface of the relevant AOI and the mean dataflow being expected. But if the surface of each AOI is well known, it is not the case for the relevant dataflow: a single bit representing the state change of a binary status indicator may be more relevant than the compressed data representing the differences between two frames of the video. So that kind of normalization was not made. Moreover, the estimation of the symbolic state of one subject is achieved by comparison between the subjects and not within each subject. This kind of comparison is the key for the definition of the domain functions and not only for the PCV. The PCV depends on the period of time



(a) Reference time 1 s.

(b) Reference time 10 s.

Figure 6.2: PCV histogram.

taken into account. If we take as the reference time "1 s" the percentage of time

spent on the video will pass from video0% to video100% in "1 s", if we assume that the mean time spent on each AOI is greater than "1 s". The result is almost a binary value (see figure 6.2a) telling us if the participant is now watching AOI 8 but one second late. On the other hand taking as the reference time "1 minute" will give us a richer distribution, but the drawback is a delay of the order of "1 minute". We have chosen a trade-off, i.e. the shortest period for which the sum of the video0% and video100% cases is less than 15% of the whole distribution. That gives us a 10 s period (see figure 6.2b).

Number of AOIs (NBAOI) and Switching rate (SWR) Two other complementary metrics come from the ET data: the number of AOIs being scanned in a defined amount of time, and the numbers of changes of AOIs in a defined amount of time. NBAOI allows us to estimate the part of the whole available dataflow being actually caught by the human. SWR estimates the rate at which the information is updated by the human. They represent respectively a measure of the situation awareness completeness and of the situation awareness obsolescence. As done for the PCV the period has been chosen as the shortest for which the sum of 0% and 100% cases is less than 15% of the whole distribution. That gives us a 10.5 s period for the SWR and 20 s period for the NBAOI.

6.4.2 ECG data

Based on previous literature [MA07], we collected cardiovascular measures in order to estimate the workload and stress the participants had experienced. We collected Heart Rate (HR) time series with 8Hz sampling. The HR was filtered using a 5-second sliding average window (HRF).
In order to estimate the anxiety the HR irregularity is also used [BR69]. The metrics used is the HR standard deviation (HRS). As reference time we use 5 s.

6.5 Data aggregation through fuzzy rules

It is appropriate to use fuzzy logic when a mathematical model of the phenomena does not exist, when the input signals are noisy, when variables are continuous, when it exists an intuitive relation defined in terms of natural language between the input and the output [MA07], [Cox92]. All those conditions are true for the attentional tunneling characterization problem, so we have decided to use fuzzy logic to aggregate the data. Figure 6.3 shows the organization of the data aggregation

system, which is described in the next section.

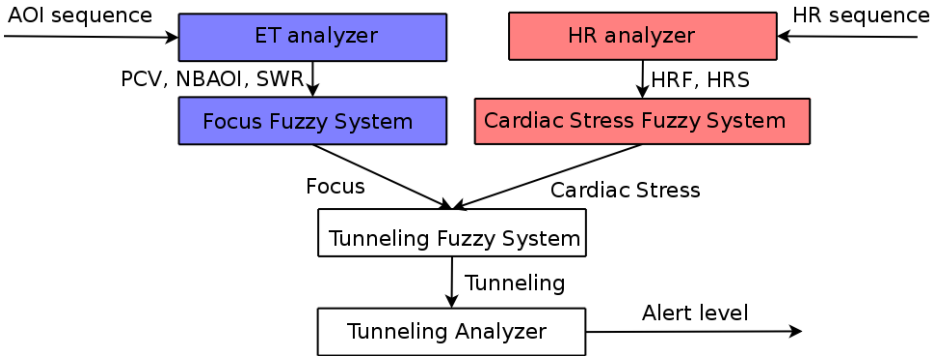


Figure 6.3: Data aggregation system.

6.5.1 Data aggregation

The AOI sequence is analyzed in order to estimate the current PCV, NBAOI and SWR for each frame (24Hz). A first fuzzy box synthesizes this information and outputs what we call Focus (F). Focus is a measure of how much the participant is focusing on the video, and how much their situation awareness is incomplete and obsolescent.
HR sequence is analysed in order to estimate the current HRF and HRS for each frame (8Hz). Cardiac Stress (CS) is a synthesis of both signals. The synthesis is made *via* a second fuzzy box.
The last step is the synthesis of both F and CS. CS has been oversampled to reach 24 Hz *via* a zero-order hold. We call Tunneling (T) the final output. T is interpreted as a symbolic state. Hereafter we use a three-level tunneling alert representation. In order to have an alert level that is blind to little oscillations of T near the level transition boundaries, those boundaries are sense-dependent, i.e. level A to level B transition boundary (engage level B) is greater than level B to level A (disengage level B) one. More into the detail :

L12, Level “alert 1” to Level “alert 2”: $T > 0.45$

L21, Level “alert 2” to Level “alert 1”: $T < 0.40$

L23, Level “alert 2” to Level “alert 3”: $T > 0.60$

L32, Level “alert 3” to Level “alert 2”: $T < 0.55$

6.5.2 Fuzzy rules

Domain functions

As we have said the estimation of the symbolic state of one participant is achieved by comparison between the participants and not within each participant. Indeed considering each single participant would lead us to define the values "high", "medium" and "low" just in the frame of the relevant metrics time series: with the within evaluation it is impossible to have a time series all the time on "mean" values, i.e. emphasis would always be put on the minimum and maximum even if they are, in comparison to other subjects, relatively near one to another.
For instance consider the case of the HRF. For the participant A HRF values are always within the boundaries $[49, 51]$. If we stay within this time series we should define (for this participant) values equal to 51 as "high". For participant B the HRF value is always within the boundaries of $[40, 60]$, and for participant C and D as well. If we merge the four time series we could consider values equal (for instance) to 57 as "high" for all the participants: in this case for participant A HRF is "medium" for the whole time series, as we want[2].
For that reason a random 5-subject sample has been taken in order to define a unique set of domain functions valid for the 12 participants. Those domain functions have been used to validate the model using the remaining 8 subjects as tests. The domain functions for PCV, SWR and NBAOI have been defined as follows:

- Low if under the 25th percentile of the sample
- Medium if on the 50th percentile of the sample
- High if over the 75th percentile of the sample

For the Low value (Medium value) in the range 25th/50th a linear interpolation from 1 to 0 (0 to 1) has been performed. The same kind of interpolation has been performed in the range 50th/75th for the Medium/High values. For HRF and HRS the domain functions are as follows: Low if on the minimum value, High if on maximum value. For the Low value (High value) a linear interpolation from 1 to 0 (0 to 1) has been performed.

Rules

The rules that have been written are the result of a learning process and are a representation that is as close as possible to the relationships found in the literature (see

[2] In the example all the participants' HRF time series have the same arithmetical mean: 50. That is not the case for the actual HRF time series: an nondimensionalization using the mean has been performed in order to override this problem.

section 6.4). The expression of the rules in natural language is the following: for the Focus rules we have stated that it is directly proportional to PCV and inversely proportional to NBAOI and SWR. For the Cardiac Stress the rule is that if both HRF and HRS metrics give the same information then there is no doubt about the CS. Otherwise the synthesis is uncertain and will give a medium level output. As for the Tunneling rules we have stated that the level of T is the greatest whenever F or CS is high, as a worst case rule[3]. Here is the list of the encoded rules:

Focus rules

if (PCV is high) then (F is high)

if (PCV is medium) then (F is medium)

if (PCV is low) then (F is low)

if (NBAOI is high) then (F is low)

if (NBAOI is medium) then (F is medium)

if (NBAOI is low) then (F is high)

if (SWR is high) then (F is low)

if (SWR is medium) then (F is medium)

if (SWR is low) then (F is high)

Cardiac Stress rules

if (HRF is low) and (HRS is low) then (CS is low)

if (HRF is high) and (HRS is high) then (CS is high)

if (HRF is high) and (HRS is low) then (CS is medium)

Tunneling rules

if (F is low) and (CS is low) then (T is low)

if (F is low) and (CS is medium) then (T is medium)

if (F is low) and (CS is high) then (T is high)

if (F is medium) and (CS is low) then (T is medium)

if (F is medium) and (CS is medium) then (T is medium)

[3]However for real time further application it is preferable to have false negative tunneling rather than false positive tunneling.

if (F is medium) and (CS is high) then (T is high)

if (F is high) and (CS is low) then (T is high)

if (F is high) and (CS is medium) then (T is high)

if (F is high) and (CS is high) then (T is high)

6.6 Results

Figures 6.4a and 6.4b show the results for two subjects previously labelled respectively as "attentional tunneling" and "Ok, conflict perceived", indeed for all 12 subjects it is well known through direct observation of the experimenter whether they had experienced attentional tunneling or not, so this knowledge is used in order to evaluate the output of the model.

Time references on the figures are:

P1: start of phase "research area"

P2: start of phase "search target"

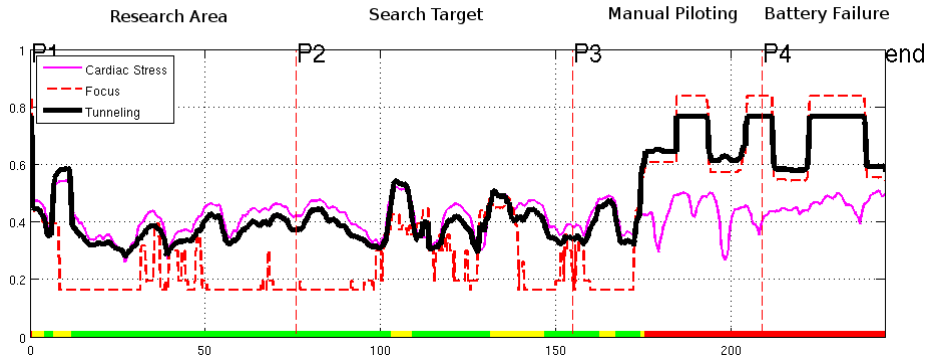
P3: start of phase "identify target" (i.e. manual piloting)

P4: failing battery alarms, piloting mode "supervised", start of the conflict

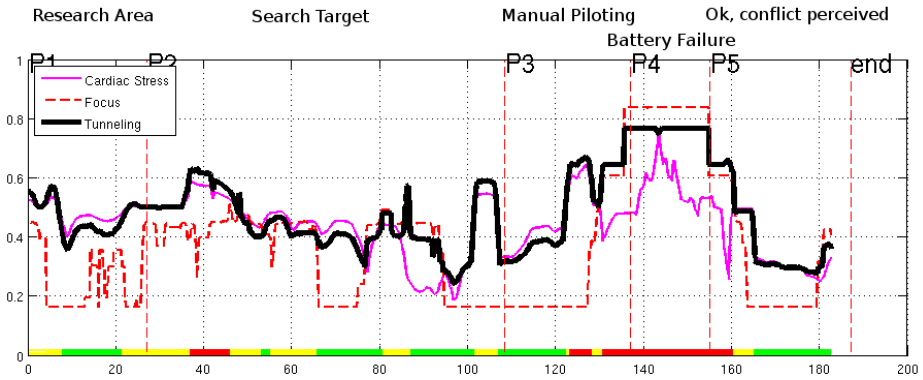
P5 (if present): observed end of the conflict [MTD10b].

On the time axis the alert level is represented by a three- color code (red, yellow and green in colored version, black, dark grey and light grey in black and white version). On the figures the red dashed line represents the Focus (output of the first fuzzy box), the pink line represents the Cardiac stress (output of the second fuzzy box), and the black line represents the Tunneling (output of the last fuzzy box, that takes the Focus and the Cardiac stress as inputs).
What characterizes the participant "conflict perception" for our model is the Tunneling decreasing to "low" values within 50 seconds from the "battery failure" event triggering (reference P4).

As we can notice for case A during manual piloting (from reference P3 to P4), the alert level goes from "low" to "high". The alert level is stable on "high" for the rest of the mission. This is a recurrent trend for many participants: their commitment in piloting makes them focus on the video AOI, scan less and less frequently others AOIs and increase their cardiac stress. This is only a general trend: for instance the cardiac stress for case A does not increase dramatically. As we can



(a) Human's "state" during the mission. Case A: attentional tunneling.



(b) Human's "state" during the mission. Case B: Ok, conflict perceived.

notice for case A the conflict is not perceived (i.e. Tunneling on "high" value from reference P4 till the end of the mission 50 seconds later).

As for case B, the alert level goes from "low" to "high" (respecting the recurrent trend) during manual piloting. After the start of the conflict the alert level is stable on "high". As we can notice for this participant Tunneling decreases to "low" value within 35 seconds from reference P4, more precisely 10 seconds after the end of the conflict (reference P5). Moreover a significant reduction in the Tunneling value is visible exactly starting from reference P5, confirming the soundness of the Tunneling metrics.

For case A the longer period on alert level high before P4 may be a precursor for the subsequent attentional tunneling after P4. To have such a kind of precursor will be useful in further real time applications in order to reduce the delay between

the occurrence of attentional tunneling and the remedy (i.e. replanning, changes in authority sharing [MTD10b], countermeasure sending [DTCR09b]).

6.7 Discussion

This study confirmed that the occurrence of a conflict during mission management is a precursor to the degradation of human operator-automation interactions as stated in the Introduction. The behavioural results showed that a majority of operators (8 out of 12) persevered to achieve the no-longer-relevant identification task, despite the three different items of information displayed on the GUI dedicated to alerting them. The particular behaviour of the robot, that started to roll away on its own as soon as the joystick was released, provoked typical "automation surprise" situations [SWB97] and led most participants to continuously take over in order to drive the robot close to the target. Only four participants (i.e., 30.8%) perceived and understood the origin of the conflict and then decided rapidly to let the robot go back to base. This is testimony to the robustness of perseveration behaviour, and stresses the importance of understanding the reasons that lead the human operator to perseverate, and the factors that contribute to the adoption of the appropriate behaviour when dealing with a conflict. These results demonstrated that it was a key issue to detect such impaired cognitive state.
Our formal approach tends to show that it is possible to infer the occurrence of attentional tunneling using fuzzy rules. The latter are consistent with literature about attentional tunneling. Indeed, they highlight that attentional tunneling was related with a strong reduction of the number of areas of interest (NBAOI) and a decrease in switching rate (SWR) which correspond respectively to fewer scanned areas of interest on the user interface and a decreased saccadic activity . Furthermore, the heart rate (HR) indicator confirms that this narrowing of the visual field on specific sources is associated with higher cardiac activity.
Although our results are promising for detecting attentional tunneling, it has several limitations. First, the efficiency of our algorithms remains limited considering domains of applications such as aviation or unmanned vehicles where safety is critical. Indeed, a challenge of our research is to design real time algorithms that automatically trigger countermeasures and a lack of reliability could lead to trigger spurious cognitive countermeasures. Such an intervention should be considered as a last resort when the other traditional alerts have proven to be inefficient to cure attentional tunneling. Another concern with our study is that the fuzzy rules linking inputs (e.g. the heart rate used as a psychological stress indicator) and the output (e.g. "the level" of attentional tunneling) were set *a priori* from expertise. A consistent way to avoid such a drawback is to use automated Machine Learning

techniques [RDR$^+$]. Moreover the metrics result from the analysis of the operator's gaze related to the AOIs on the interface, which requires the expert knowledge of the interface. Consequently, we would like to provide generic metrics that would be interface-independent and thus could be extended to other domains.

Chapter 7

Reversibility

7.1 Introduction

Whether by omission or commission, mistake or slip [Rea90], errors are an everyday part of human existence. While it is possible to design systems to be less vulnerable to errors, the systems also require the ability to gracefully recover from errors when they do happen. An analysis of the ease of recoverability could be very useful in the evaluation of designs, particularly for the design of safety critical systems. While a complete assessment of recoverability is beyond the current scope, this chapter will focus on the reversibility of commands as one dimension of a recoverability metric. Specifically this chapter will describe a method for formally identifying actions that are not reversible within one step, or are irreversible. *Reversibility* [CCH08] is generically meant as the property to undo the effects of some action after the execution of a sequence of actions. When defining the reversibility property, it is important to define the domain of interest that is relevant for the designer's purposes. If time is an explicit state variable, all the actions are non reversible. The same is true if some of the state variables are monotonous functions of time. Using aviation as an example domain, there are many action sequences that are not reversible due to the passage of time (e.g. the fuel level always decreases, distance to destination should decrease, etc.) In the same way, there are aircraft automation modes that are irreversible due to the time dynamics. In this chapter a five-level reversibility scale is defined for human's action-driven state change. One reversibility property chapter correspond to each level. It will be shown that the set of state changes verifying higher level properties will be included in the sets of state changes verifying lower level properties. An automated test to assess the reversibility degree of each of the human-automation state changes is formalized, which can verify three out of five properties. The check of the last two

properties is out of the scope of this chapter.

7.2 Definitions

7.2.1 Reversibility

Definition [Reversibility]: a state change of a system from state z0 to state z1 triggered by action 1 is said *reversible* if from state z1, it exists at least one sequence of actions that at some point in the future will take back the system to the initial state z0 (from [CCH08]), see figure 7.1.

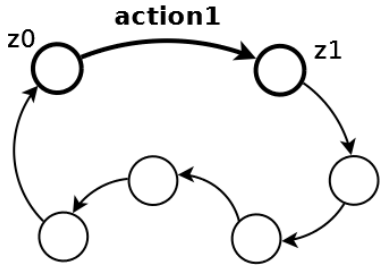


Figure 7.1: Reversibility

7.2.2 Undo

Definition [Undo]: The *undo by a specific action*, inspired by [CCH08], is a special case of the general reversibility definition: the sequence of actions is replaced by a unique action, always the same independently from the action to undo (see figure 7.2).

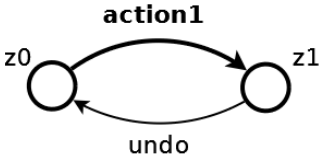


Figure 7.2: Undo

The *undo* is desirable for non critical domains, as for instance text editor software. Nevertheless, even for this application, predictability issues arise: *despite*

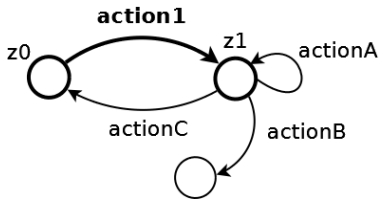


Figure 7.3: One action reversibility

the general agreement of the importance of undo, few systems supply more than the simplest single-step undo command and, even then, the effect of command and when it can be applied is often far from obvious. In a recent study of the use of Microsoft Word, it was found that even expert users were unable to both predict the effect of the undo command or recognise its behaviour. Undo support for single user systems is regarded as essential, but recognised to be fraught with potential pitfalls [AD91].

In the literature the emphasis is put on action triggered state changes whose consequences, for the same action, are dependent on the state of the system. That kind of state change falls into the general definition of "inconsistent behaviour" [LPS$^+$97, CCH08] or also "moded behaviour" [Fea05, Fea07] and is widely recognized as a vulnerability. Note that the *undo* may have different consequences depending on which was the last performed action to undo. For that reason it is an "inconsistent behaviour".

7.2.3 One action reversibility

For safety critical domains such as aeronautics predictability is highly desirable [Fea05]. So the *undo*, because of its predictability issues [AD91, LPS$^+$97, CCH08, Fea05], is not a desired feature in the aeronautical domain. The absence of the *undo* function does not lead to the loss of reversibility, not even the loss of the one action reversibility. It is enough to show that any action can be undone simply by a proper re-action.

Definition [One action reversibility]: a state change of a system from state z0 to state z1 triggered by action 1 is said *one action reversible* if from state z1, it exists at least one single action that will take back the system to the initial state z0 (inspired by [CCH08]), see figure 7.3.

Example: Consider a system with just three state values *high/medium/low* and three actions *up/down/undo*. The initial state is *medium*. After performing an *up* action the new state is *high*. To reverse the effects of this action it is possible to perform the *undo* action. To obtain the same result it is also possible to execute the *down* action. In the first case *the effects of one action via a specific action (undoing) are nullified*, in the second one *the effects of one action are reversed via a relevant reaction (reversing in one action)*.

7.2.4 Totally unrecoverable state change

Definition [Irreversibility]: The execution of an action from a state z0 can lead to a state z1 from which there is no possible way back to z0. Such a state change, in accordance to definition [Reversibility], is *irreversible*.

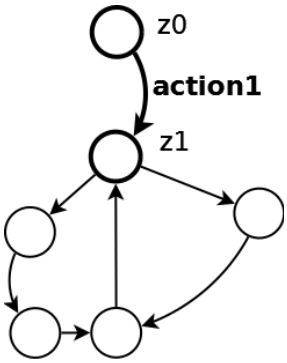


Figure 7.4: Irreversibility

Definition [Total unrecoverability. 1]: The execution of an action from a state z0 can lead to a state z1 from which there is no possible way out, i.e. there is no further sequence of actions that can lead to a new state that is different from z1. z1 is a *blocking state* (figure 7.5). Such states could correspond to a physical limit of the system or a system failure. They can also be the result of a design error: this is typically the case of system deadlocks after the execution of an unexpected sequence of actions. Those state changes are said to be *totally unrecoverable*.

Example: let us consider a system with a four-value state variable *high/medium/low/out of order* and three possible actions *up/down/put out of order*. Initial state is *medium*. After the execution of the action *put out of order* the state of the system becomes *out of order*. At that point there is no possible sequence of actions to change state (in this simplified model there is no *repair* action).

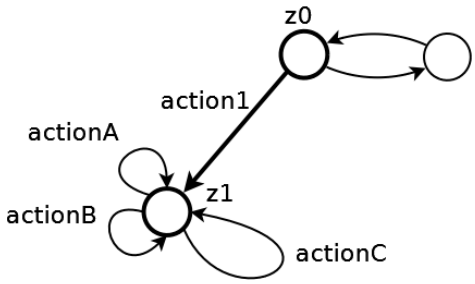


Figure 7.5: Total unrecoverability

Another definition that is equivalent to the previous one is:

Definition [Total unrecoverability. 2]: a state change triggered by an action 1 from a state z0 resulting in the new state z1 is said *totally unrecoverable* if from z1 there is no action that can lead to a state that is different from z1.

This second definition is easier to verify because it only needs the evaluation of the system state after the execution of just one action. Therefore it will be used in the algorithm to verify the *totally unrecoverable* property.

Definition [Eventually totally unrecoverable state change]: a state change may lead to an *impasse state* from which any possible action leads to a *blocking state* or to another *impasse state* (figure 7.6). This state change is called *eventually totally unrecoverable*.

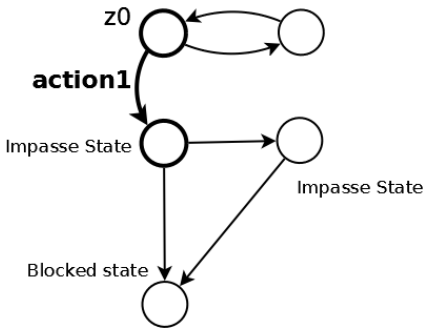


Figure 7.6: Impasse state and blocked state

The detection of *eventually totally unrecoverable state changes* needs an iterative process.

7.2.5 Reversibility scale

Different types of reversibilities and irreversibilities have been defined in the previous sections. Table 7.1 summarizes as a scale the different cases of reversibility from the "most reversible" to the "less reversible".

Name	Property
Undo	A particular action to come back to z0
One action reversible	An action to come back to z0
Reversible	A sequence of actions to come back to z0
Irreversible	No sequence of actions to come back to z0
Eventually totally unrecoverable	Any sequence of actions leads to a blocked state
Totally unrecoverable	No action to leave z1

Table 7.1: Reversibility scale

If we note as $\subset$ "is a particular case of", we get the following relations:

Undo $\subset$ *One action reversible* $\subset$ *Reversible*

Totally unrecoverable $\subset$ *Irreversible*

Combining them in just one formula:
$\neg$ *Undo* $\supset \neg$ *One action reversible* $\supset$ *Irreversible* $\supset$ *Totally unrecoverable*

or also:

Undo $\subset$ *One action reversible* $\subset$ *Reversible* $\subset \neg$ *Totally unrecoverable*

7.3 An automated test for reversibility assessment

There are many formal models of human-automation interfaces, however the vast majority of them are computationally equivalent to a finite state transition system or finite state machine [BBS12]. Among such models ADEPT (Automation Design and Evaluation Prototyping Toolset) [1] [Fea05, Fea07] performs a set of automated analyses on the structure of the human-machine interface in order to verify properties like *completeness* and *consistency* properties in addition to vulnerability

[1] We worked with ADEPT during a visit at NASA Ames from March to May 2012.

checks detailed hereafter. However it does not perform any reversibility verification [CCH08]. Our contribution is to define an additional algorithm within ADEPT so that it can evaluate a set of reversibility properties.

7.3.1 ADEPT

ADEPT checks a number of properties that might flag potential vulnerabilities of the interaction:

Moded input: the same action performed by the user has many possible effects, depending on the state of the system.

Armed behaviour: the effects on the system of an action of the user may be delayed.

Automated behaviour: a system state change that does not need an action of the user to be triggered.

Inhibited behaviour: an action of the user that has no effect on the system state.

Similar feedback: the same display is used for more than one behaviour of the system.

In ADEPT state transitions are represented as triples (input state, user action, output state). If the state change does not need a user action to be triggered, the triplet will be noted (input state, "no action", output state).

Definition [Situation]: a *situation* is defined as the conjunction between a proposition about the *actions* and a proposition concerning the *input states*: *actions* are described as a disjunction of actions and input states are described as a conjunctive normal form, i.e. a logic conjunction between the state variables and a disjunction of values of the same variable.
Actions and *input states* are either defined explicitly or take the parametric value "no matter which value/no matter which action (**)".

Definition [Behaviour]: a *behaviour*, which is the result of a situation, is defined as a logic conjunction between state variables that take just one value at a time. This value is either defined explicitly or take the parametric value "same as input (*)".

Example:

$$situation : (action = [a_1 \vee a_3]) \wedge (x_1 = [v_{11} \vee v_{12}]) \wedge (x_2 = [**])$$

$$behaviour : (x_1 = [*]) \wedge (x_2 = [v_{22}])$$

In this example the situation is expressed in natural language as: "action is either a_1 or a_3 and variable x_1 value is either equal to v_{11} or v_{12}, variable x_2 takes any value". The behaviour is expressed as: "variable x_1 value is the same as input and variable x_1 value becomes v_{22}".

Definition [Logic table]:
The set of pairs (situation, behaviour) is represented in ADEPT in a compact table called *logic table*. The first column of the table contains actions and state variables names, the second column contains actions and state variables values. From the third column, each column represents a pair (situation, behaviour), consequently pair number 1 will be represented in the column called c1. In those columns an empty box is set to 0 (or false). If for some *situation* all the boxes corresponding to the actions or a state variable are empty, the action or variable takes [**]. If for some *behaviour* all the boxes for a state variable are empty, the variable has the same value as the input [*].

Example (see table 7.2):

The behaviour of a system "three level variable control" is shown in table 7.2. The user's actions have effects on variable X and on the state of the keyboard (action *Ops!* represents the accidental breakage of the keyboard).
The logic table columns (c1 to c8) representing the pairs (situation, behaviour) are as follows:

- from c1 to c6: user increases or decreases X.
- c7: accidental breakage of the keyboard, from this point it is out of order.
- c8: if the keyboard is out of order, any user action has *no effect* (i.e. all the variables keep the same value as the input).

Regarding columns c3, c4 and c8 their behaviour is *no effect*. Without those columns the logic table would not be complete.
For instance, column c8 represents the following pair (situation, behaviour):

$$situation : (action = [**]) \wedge (X = [**]) \wedge (keyboard = [Out\,of\,order])$$

$$behaviour : (X = [*]) \wedge (keyboard = [*])$$

		c1	c2	c3	c4	c5	c6	c7	c8
SITUATION		Increase X starting from Low	Increase X starting from Medium	Increase X starting from High	Decrease X starting from Low	Decrease X starting from Medium	Decrease X starting from High	Accidental breakage of the keyboard	Any action in case of keyboard out of order
Actions									
	Increase X	1	1	1					
	Decrease X				1	1	1		
	Ops!							1	
	No action								
State									
Keyboard	Ok	1	1	1	1	1	1	1	
	Out of order								1
X	High			1			1		
	Medium		1			1			
	Low	1			1				
BEHAVIOUR		X effectively increases	X effectively increases	no effect	no effect	X effectively decreases	X effectively decreases	Keyboard becomes out of order	no effect
State									
Keyboard	Ok								
	Out of order							1	
X	High		1(ok)	1(ok)					
	Medium	1(ok)	1(a)				1		
	Low	1(c)		1(b)		1			

Table 7.2: Logic table for the system “three level variable control”

The natural language description of this pair (situation, behaviour) is *if the keyboard is out of order any action for any value of X has no effect*.
For more details on the construction of a logic table see [Fea05, Fea10].

Definition [complete logic table] A *logic table* is *complete* if each combination (input state, action) is listed in at least one *situation*.

Definition [input consistent logic table] A *logic table* is *input consistent* (or simply *consistent*) if each combination (input state, action) is listed in at most one *situation*.

Consequently, in a *complete and consistent logic table*, for any combination (status input, action) there is always one and only one *situation*, and one resulting *behaviour*. Note that in a *complete logic table* any combination (status input, action) must be explicitly listed in the table, including those whose behaviour is *no effect*.

7.3.2 Towards reversibility assessment with ADEPT

As in ADEPT each pair (situation, behaviour) may represent several state changes at the same time (e.g.: $situation : (action = [a_1]) \wedge (X = [v_1 \vee v_2])$, $behaviour : (X = [v_3])$ represents the transitions $t_1 : action = [a_1] \wedge X = [v_1] \rightarrow X = [v_3]$ and $t_2 : action = [a_1] \wedge X = [v_2] \rightarrow X = [v_3]$) the formal definitions of reversibility need to be adapted, according to a conservative choice: a pair (situation, behaviour) satisfies a given reversibility property when all the state changes represented by the pair satisfy this property. In the same way a pair (situation, behaviour) satisfies a given irreversibility property if at least one state change represented by the pair satisfies this property. See Table 7.3 for definitions of reversibility properties suitable for pairs (situation, behaviour).

Property	State changes of a pair that must verify the property
Undo	All of them
One action reversibility	All of them
Reversibility	All of them
Irreversibility	At least one
Eventually totally unrecoverable	At least one
Totally unrecoverable	At least one

Table 7.3: Reversibility scale for pairs (situation,behaviour)

As shown in table 7.1 the verification of the *reversibility* property needs the evaluation of the effects of an unknown-length sequence of actions. In this algorithm we have decided to evaluate the effects of one action and one further action to reverse the effect of the first one. Consequently we will check the *one action reversibility*, *eventually totally unrecoverable* and *totally unrecoverable* properties. The ADEPT algorithm for reversibility assessment is as follows:

Prepare a list of all the combinations of the possible state variable values, each of them will be a called *state instance*.

For every pair *(situation, behaviour)* (i.e. for every column of the logic table) verify if any of the *state instances* is a valid input state for this pair, i.e. is compatible with the relevant *situation*. In this case this *state instance* is called z0. Evaluate the resulting output state, called z1.

For this state z1 verify which other pairs have a *situation* compatible with it. We call z2 the resulting output state for this pair (a different z2 for each pair).

Note that for each pair there are many initial *state instances* z1, and for each z1 many possible applicable pairs (and for each of them a final *state instance* z2).

Then we classify *each pair* as follows:

If for at least one z1 of this pair (see Table 7.3) all the z2 are identical to z1, the pair is classified as *Totally unrecoverable*, and z1 is classified as *blocked state*. The information about z1 and the relevant initial state z0 are available.

If for at least one z1 of this pair all the z2 are different from z0, the pair is classified as *Non one action reversible*. The information about z1 and the relevant initial state z0 are available.

If for all the z1 of this pair at least one z2 is identical to z0, the pair is classified as *One action reversible*. The information about z1 (all of them) and the relevant initial state z0 are available.

Iterating:

If for at least one z1 of this pair all the z2 are identical to a state classified as *blocked* or *impasse*, the pair is classified as *Eventually totally unrecoverable*, and z1 is classified as an *impasse state*. The information about z1 and the relevant initial state z0 are available.

Reiterate on the whole set of pairs until there are no new states left classified as *impasse state*.

7.3.3 Example: three level variable control

Let us consider again the example described by table 7.2. Note that each c_i-labelled column of the table corresponds to a pair (situation,behaviour), for instance column c7 represents pair number 7 (situation: Actions/Ops!,State/Keyboard/Ok,State/X/any value, behaviour: State/Keyboard/Out of order, State/X/same as input). First the automatic analysis is performed on a correct design. Afterwards it will be performed on three different design error scenarios in order to assess their impact on the reversibility of the system.

Correct design: "1(ok)" values in columns c1, c2, c3.
The results of the analysis are:

```
pair number 1 is 1-action reversible all of its input states
pair number 2 is 1-action reversible all of its input states
pair number 3 has no effect
pair number 4 has no effect
pair number 5 is 1-action reversible all of its input states
pair number 6 is 1-action reversible all of its input states
pair number 7 is totally unrecoverable
for input state: 1  0  1  0  0
for input state: 1  0  0  1  0
for input state: 1  0  0  0  1
pair number 8 has no effect
```

Not surprisingly pairs number 1, 2, 5 and 6 are found to be 1-action reversible for all their input states: they describe the action of increasing or decreasing the value of variable X.
Pair number 7 is found to be totally unrecoverable for three input states. Those input states are expressed in the form of an array that has the same notation as the input state in the logical table (where the zeros are omitted): array [1 0 1 0 0] represents state [Keyboard (Ok), X (High)], array [1 0 0 1 0] represents state [Keyboard (Ok), X (Medium)] and array [1 0 0 0 1] represents state [Keyboard (Ok), X (Low)]. This is coherent with the expected behaviour: putting the keyboard out of order (represented by column c7 that has as input state [Keyboard (Ok), X (any value)]) should be modelled as the only totally unrecoverable state change for the user operational domain.

Pairs number 3, 4 and 8 are found to have no effect (in accordance with their definition on table 7.2) and are not classified on the reversibility scale as they have no effect to be reversed.
For the next results, the algorithm raw outputs will be skipped.

Design error "1(a)" (column c2): action *increase X* in the case of *X/Medium* has no effect. The result of the automatic analysis is the same as in the nominal case, with the exception of pair 6 that is classified as:

```
NON 1-action reversible
```

Indeed column c6 represents the transition from *X/High* to *X/Medium*. Due to the fact that the transition from *X/Medium* to *X/High* is corrupted, c6 has become *non reversible* and therefore *non one action reversible*. The automatic analysis cannot recognize *non reversible* state changes but it recognizes those that are *non one action reversible*, so the column is recognized as "just" *non one action reversible*.
Regarding c7 nothing has changed in the design and the analysis results are the same.

Design error "1(b)" (column c3): the action *increase X* in the case of *X/High* leads to the value *X/Low*. The result of the automatic analysis is the same as in the nominal case, with the exception of pair 3 that is classified as:

```
NON 1-action reversible
```

Indeed two actions *increase X* are needed to restore the input state.

Design error "1(c)" (column c1): the action *increase X* in the case of *X/Low* has no effect. The result of the automatic analysis is the same as in the nominal case, with the exception of pair 5 that has become: :

```
eventually totally unrecoverable
```

for input state [Keyboard (Ok), X (Medium)]. Column c5 represents the transition from *X/Medium* to *X/Low*. Due to the fact that the transition from *X/Low* to *X/Medium* is corrupted, c5 has become *eventually totally unrecoverable*: after the execution of c5 there is no other possible state change than c7, that leads to a blocking state (i.e. it is no more possible to change X but it is still possible to put the keyboard out of order). So c5 leads to an impasse state.

In all the error scenarios the results of the analysis highlight an incoherence with the expected behaviour of a correct design. In the nominal case we expected

only one pair to be totally unrecoverable, all the others one-action reversible. But because of the introduction of the errors some differences from the expected behaviour arise. Those differences could alert the designer about the presence of errors.

7.3.4 Example: a simplified autopilot model

A "Go-around" is an aborted landing of an aircraft that is on final approach. Many modern aircraft fly-by-wire systems include a "Go-around mode" that automatically sets the throttle to the maximum level. Moreover, depending on the manufacturer, it either switches off the autopilot or it just switches off the instrument landing system mode. The Go-around mode is designed to help the crew to quickly increase thrust and abort a landing. Reversing a Go-around decision can be hazardous. Autopilot systems are often designed in order to require many steps to disengage a Go-around mode in order so as the pilot to be fully conscious of the new state of the autopilot.

An autopilot mock-up model is presented in this section in order to analyze the reversibility of the Go-around mode engagement. For the behaviour of the system see the logic table 7.4.

In this simplified model a limited number of state transitions are modelled. The system state variables are autopilot modes (Mode 1, Mode 2, Mode 3, Go around), throttle level (+, ++, +++), autopilot state (On, Off), autopilot engagement conditions (satisfied, not satisfied). Columns 5, 6 and 7 represent autopilot mode changes. Columns 9 to 14 represent thrust level changes. Columns 2 and 3 represent autopilot state changes. Column 2, unlike pair 3, involves also the autopilot mode: when disengaged the autopilot is set on Mode 1. Column 1, corresponding to the engagement of mode "Go Around", changes both autopilot mode and throttle level. Note that column 1 represents 48 state transitions at the same time (48 transitions that have as input the cartesian product of 4 autopilot modes, 3 throttle levels, 2 autopilot states and 2 autopilot engagement conditions).

The results of the automated analysis are as follows.
Pairs number 1, 2 and 3 are found to be

```
NON 1-action reversible
```

Pairs number 4, 8, 11 and 14 are found to have (in accordance with their definitions in table 7.4)

```
no effect
```

Pairs number 5, 6, 7, 9, 10, 12, 13 and 14 are found to be

		c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14
SITUATIONS		Button Go around pressed	Button Autopilot Off pressed	Button Autopilot On pressed	Button Autopilot On pressed, engagement criteria not satisfied	Mode 1 selection	Mode 2 selection	Mode 3 selection	Mode Autopilot 1, 2, 3 selection when Go-around mode engaged	Throttle level from + to ++	Throttle level from ++ to +++	Increase throttle level when already at +++	Throttle level from +++ to ++	Throttle level from ++ to +	Decrease throttle level when already at +
Pilot actions															
Actions, button pressed	Go-Around engaged	1													
	Mode 1 selection					1			1						
	Mode 2 selection						1		1						
	Mode 3 selection							1	1						
Actions, Throttle level	Increase throttle level									1	1	1			
	Decrease throttle level												1	1	1
Autopilot engagement	On selection			1	1										
	Off selection		1												
No action															
State															
Autopilot mode	Mode 1					1	1	1							
	Mode 2					1	1	1							
	Mode 3					1	1	1							
	Go Around								1						
Throttle level	+									1					1
	++										1			1	
	+++											1	1		
Autopilot state	On														
	Off														
BEHAVIOUR		Go-Around engaged	Disengage autopilot	Engage autopilot	No effect	Mode 1 engaged	Mode 2 engaged	Mode 3 engaged	No effect	Throttle level from + to ++	Throttle level from ++ to +++	No effect	Throttle level from from +++ to ++	Throttle level from ++ to +	No effect
State															
Autopilot mode	Mode 1		1			1									
	Mode 2						1								
	Mode 3							1							
	Go Around	1													
Throttle level	+													1	1
	++									1			1		
	+++	1									1	1			
Autopilot state	On			1											
	Off		1												
* the autopilot disengage le reset autopilot mode to mode 1.															

Table 7.4: Part of the logic table for a simplified autopilot model

```
1-action reversible
```

Regarding pair 1 (“Go Around” mode activation) it is not reversible in a single action because it changes the values of several state variables at the same time, when almost all other columns act only on one variable at a time. For example in the case of initial state:
[autopilot mode (Mode 3), autopilot state (On), throttle level (+)],
“Go Around” mode engagement brings the system to state:
[autopilot mode (Go Around), autopilot status (On), throttle level (+ + +)]
and the initial state cannot be recovered in less than five actions: Off selection, On selection, Mode 3 selection, Decrease throttle level, Decrease throttle level.

Pairs 2 and 3 non 1-action reversibility is due to the fact that pair 2 changes not only the autopilot state but also the autopilot mode. For example in the case of initial state:
[autopilot mode (Mode 3), autopilot state (On), throttle level (+)],
the “Off selection” brings the system to state:
[autopilot mode (Mode 1), autopilot status (Off), throttle level (+)]
and the initial state can not be recovered in less than two actions: “On selection” and “Mode 3 selection”.

7.4 System and objective situation awareness, an ADEPT representation

The logical table representation, which is designed to analyze the properties of a system and its reaction to the operator’s actions, can also be used for the analysis of the human/machine interaction in order to identify irreversibilities that are not only specific to the behaviour of a system but also stem from a bad interaction. For this analysis the logical table represents both the system state and the objective situation awareness (OSA) of the operator (for a definition of the OSA see Chapter XX).
In this approach the feedbacks are modelled with state variables.

The following examples show the results of the automatic reversibility analysis for two real cases.

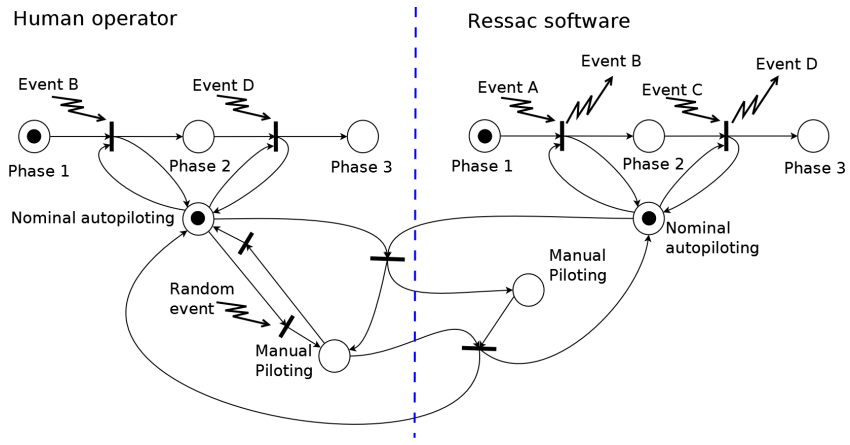


Figure 7.7: The Petri net representation of the Ressac mission

7.4.1 Example 1 : Ressac "Rain and automation"

As shown in Chapter XX, an unexpected event (the rain), and the subsequent emergency transition mode to *manual control*, leads to a blocking in the Petri net of figure 7.7. The logic table model of this mission is given in table 7.5. To build the table we took each Petri net transition and modelled it as a pair (situation, behaviour): the information on the situation is found in the transition *preconditions* (input places), the information on the behaviour in the transition *postconditions* (output places).

The results of the automated analysis are as follows.
Pairs number 1, 2 ,3 and 4 are found to be

```
NON 1-action reversible
```

Pairs number 5, 6, 7 and 8 are found to be

```
1-action reversible
```

The automatic analysis tells us that pairs 1, 2, 3 and 4 are non 1-action reversible as expected: indeed they represent mission phase achievements. The Petri net analysis described in chapter XX allowed us to detect a potential problem that escapes this analysis: the reversibility analysis cannot detect dead transitions.

7.4.2 Example 2: Emaxx mission

As we saw in Chapter XX, an unforeseen event (battery failure), leads to the blocking of the Petri net of figure 7.8. The logic table model of this mission is given in

		c1	c2	c3	c4	c5	c6	c7	c8
SITUATIONS		Automated achievement Phase 1	Achievement Phase 1 acknowledged	Automated achievement Phase 2	Achievement Phase 2 acknowledged	Manual mode emergency engagement	Manual mode emergency disengagement	Manual mode nominal engagement	Manual mode nominal disengagement
Actions									
	Switch manual piloting on (emergency)					1			
	Switch manual piloting off (emergency)						1		
	Switch manual piloting on (nominal)							1	
	Switch manual piloting off (nominal)								1
State									
Human operator OSA mission phase	Phase 1		1						
	Phase 2				1				
	Phase 3								
Human operator OSA piloting mode	Nominal autopiloting		1		1	1		1	
	Manual piloting						1		1
Ressac software mission phase	Phase 1	1							
	Phase 2			1					
	Phase 3								
Ressac software piloting mode	Nominal autopiloting	1		1				1	
	Nominal manual								1
Feedback	Phase 1 to phase 2		1						
	Phase 2 to phase 3				1				
BEHAVIOUR		Phase change and achievement feedback	OSA update	Phase change and achievement feedback	OSA update	Emergency mode engaged	Emergency mode disengaged	Nominal manual mode engaged	Nominal manual mode disengaged
State									
Human operator OSA mission phase	Phase 1								
	Phase 2		1						
	Phase 3				1				
Human operator OSA piloting mode	Nominal autopiloting						1		1
	Manual piloting					1		1	
Ressac software mission phase	Phase 1								
	Phase 2	1							
	Phase 3			1					
Ressac software piloting mode	Nominal autopiloting								1
	Nominal manual							1	
Feedback	Phase 1 to phase 2	1							
	Phase 2 to phase 3			1					

Table 7.5: Ressac mission logical table

table 7.6.

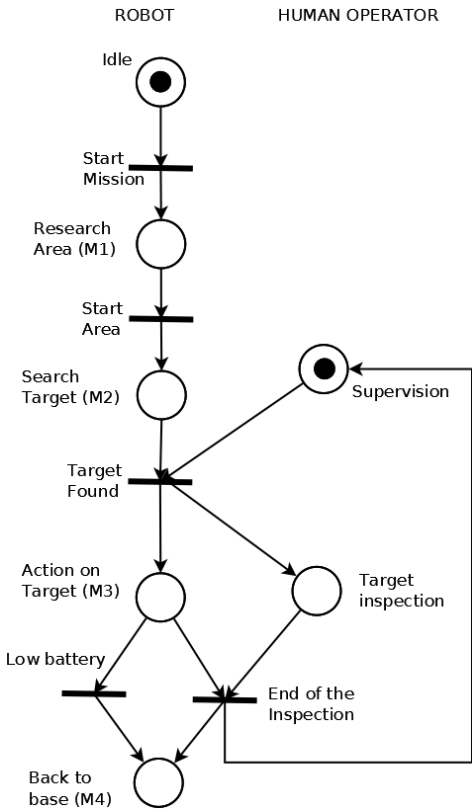


Figure 7.8: The Petri net representation of the Emaxx mission

The results of the automated analysis are as follows.
Pairs number 1, 2, 3 are found to be

```
eventually totally unrecoverable
```

Pairs number 4 and 5 are found to be

```
totally unrecoverable
```

Pair number 2 is found to be

```
totally unrecoverable
```

for input state [mission phase (M1), human operator role (Target inspection)].

		c1	c2	c3	c4	c5
SITUATIONS		Start mission	Start area	Target found	End of inspection	Battery failure
State						
Mission Phase	M0: Idle	1				
	M1: Research area		1			
	M2: Search target			1		
	M3: Action on target				1	1
	M4: Back to base					
Human operator role	Supervision			1		
	Target inspection				1	
BEHAVIOUR		Transition M0/M1	Transition M1/M2	Transition M2/M3	Transition M3/M4	Transition M3/M4
State						
Mission Phase	M0: Idle					
	M1: Research area	1				
	M2: Search target		1			
	M3: Action on target			1		
	M4: Back to base				1	1
Human operator role	Supervision				1	1
	Target inspection			1		

Table 7.6: Emaxx mission logic table

The Petri net model represents a system that evolves just in one direction with no reinitialization. All the pairs describing the early evolution of the system (pair number 1, 2 and 3) are classified as eventually totally unrecoverable as a finite sequence of actions will lead to the final state, which is a blocked state. Pairs 4 and 5 are two possible ends of the mission, and they are both classified as totally unrecoverable.
Moreover pair 2 is classified as totally unrecoverable for input state [mission phase (M1), human operator role (Target inspection)] that leads to blocked state [mission phase (M2), human operator role (Target inspection)]. It is worth noticing that with the initial marking shown in Figure 7.8 the identified blocking state is actually not reachable. Indeed the reversibility analysis evaluates the cartesian product of all the states, not only the reachable ones: it could not be otherwise because the information on the initial state is not present in the logical table representation. On the contrary only the reachable states are taken into account in the Petri net analysis.

7.5 Comparing ADEPT logic tables and Petri Nets

In the logic table representation of ADEPT, system state transitions are represented as pairs (situation, behaviour), and each pair (situation, behaviour) may represent several transitions at once. The set of transitions described in the logic table may be represented as a graph: for the construction of such a graph an exhaustive exploration of pairs (situation, behaviour) is required (for instance by means of an iterative algorithm) to define the list of transitions. For instance column c1 in logic table 7.4 represents 48 transitions (for the four state variables Autopilot mode/Throttle level/Autopilot state/Autopilot engagement conditions), 12 of them are:

1. (Mode 1/+/On/Satisfied) → (Go around/+++/On/Satisfied)
2. (Mode 2/+/On/Satisfied) →(Go around/+++/On/Satisfied)
3. (Mode 3/+/On/Satisfied) → (Go around/+++/On/Satisfied)
4. (Mode Go around/+/On/Satisfied)→(Go around/+++/On/Satisfied)
5. (Mode 1/++/On/Satisfied)→ (Go around/+++/On/Satisfied)
6. (Mode 2/++/On/Satisfied)→ (Go around/+++/On/Satisfied)
7. (Mode 3/++/On/Satisfied)→ (Go around/+++/On/Satisfied)
8. (Mode Go around/+++/On/Satisfied) →(Go around/+++/On/Satisfied)

Property	ADEPT	Petri Net
Easily model-able, small model size	+	-
State value actually reachable	-	+
Existing formal analysis	-	+

Table 7.7: Logic table and Petri net, a comparison

9. (Mode 1/+++/On/Satisfied)→ (Go around/+++/On/Satisfied)
10. (Mode 2/+++/On/Satisfied)→ (Go around/+++/On/Satisfied)
11. (Mode 3/+++/On/Satisfied)→ (Go around/+++/On/Satisfied)
12. (Mode Go around/+++/On/Satisfied) →(Go around/+++/On/Satisfied)

Therefore some formal analyses of the properties of the graph cannot be performed directly on the logic table representation.
It should also be noted that the logic table exhaustively define the cartesian product of all states, even for non-reachable states. The reversibility analysis that we have developed for ADEPT therefore recognizes irreversibilities regarding states that are actually not reachable.
On the other hand it is possible to perform automated formal verifications directly on a Petri net representation (e.g. the reachability evaluation and the dead transitions identification) but the Petri net representation is not suited to quick modelling of complex systems with a large number of state transitions (as for an autopilot).

A possible solution to design models rapidly while keeping the possibility of formal analysis is an hybrid approach in which the system is represented by means of a logical table, and a Petri net representation is computed from the logic table through the calculation of the exhaustive list of transitions input/output states as for the Go around example the formal analysis is then performed on the Petri net.
We have developed this automatic ADEPT logic table to Petri Nets converter. It explores the cartesian product of all the N input states. For each of them it computes the resulting output state thanks to the relevant pair (situation, behaviour). Those two vectors compose respectively one of the N vectors of the Pre incidence matrix and Post incidence matrix of the Petri net. The problem of the combinatory explosion of the number of vectors remains to be dealt with.

7.6 Conclusion

An algorithm to implement a reversibility check on the ADEPT logical tables has been implemented. Three possible vulnerabilities can be highlighted: the absence

of one action reversibility, the total unrecoverability and the eventually totally unrecoverablility. Note that the lack of reversibility may be either the result of a design error, or a conscious design choice, or an intrinsic lack of reversibility or the representation of a failure. The designer, once aware of those vulnerabilities, will possibly decide to modify the behaviours of the interface and estimate the impact of those changes.

The representation of the interaction we have defined is obtained by coupling the system model with a representation of the OSA (Objective situation awareness) based on the information contained in the operator procedure. This approach is not easy to implement for large systems. A different approach consists in taking into account the model of the system and the information (communication or feedback) sent to the operator. The structure of the representation of the OSA is then obtained as a copy of the structure of the system. The possible inconsistencies between both structures are due to the asynchronous exchange of information.

Chapter 8

Conclusion and further work

In this chapter we summarize our contribution, the findings and the results of the work. In the last part we propose further developments that could implement our results to real applications in the domain of human-machine conflict prevention.

Human-machine conflicts are critical situations that can create accidents. Therefore they must be studied both from a formal and experimental points of view. So as to better comprehend them and understand their effects on human-machine systems.
The work that has been carried out during this PhD is part of the research dedicated to human-machine conflicts, more precisely the work focuses on models and tools for the prevention and detection of human-machine bad interactions, especially in the aeronautics field. It is based on a multidisciplinary study at the intersection of formal methods and cognitive psychology.

8.1 Main contributions

We develop a general conflict model based on the observation of real conflict cases using Petri nets. This model is based on the identification of a conflict pattern in the model of the machine, which is described *via* the definition of internal state changes. The model is general in so far as it can detect different conflicting situations with no *ad hoc* assumptions.
An experiment in a flight simulator with pilots has been designed in order to test the soundness of the formal approach, in other words to assess whether the *a priori* identified conflict patterns indeed create conflicts between the automation and the pilot, and whether those conflicts are detected or solved. The results of the experiment tend to support the formal approach for conflict prediction but also show that conflicts that are identified without distinction by the formal analysis induce differ-

ent pilots' behaviours. Therefore experimental analyses of such situations remain necessary in order to correctly assess their criticality and understand their dynamics.

This *a priori* model is based on general assumptions, nevertheless those assumptions are *tight* as so far as we assume the human to have a very limited perception. Hence we make further assumptions of the *possibility* that the human has limited perception: we propose an enhanced model with uncertainty management to be used as a real time conflict detection and identification tool. This uncertainty model is based on general assumptions about the human knowledge of the behaviour of the machine, on the possibility that the human does not perceive feedbacks or that their initial assessment of the internal state is wrong. Those *relaxed* assumptions are used to define an error model. Some authors have already proposed error models for the human assessment of the machine, which are deterministic because they assume that the human will certainly make those errors. Our uncertainty model is original in so far as it considers as possible the correct assessment of the situation and the error model *as well* (with different degrees of plausibility). As long as the human performs actions that are coherent with the internal state the uncertainty model estimates that the human is correctly assessing the internal state. When the human performs an action that is not coherent given the actual internal state the uncertainty model finds if there is in the past history an assessment error that could explain the execution of this action. This uncertainty model has been tested on the data coming from the flight simulator experiment: the results are encouraging as the tool indeed finds explanations to incoherent actions that *seem* to be correct conflict explanations. Nevertheless this tool should be further tested in real time applications in order to verify the validity of the model estimates.

The uncertainty model could be further enriched thanks to data coming from the observation of the human thanks to dedicated sensors, and methods to infer the human's attentional *state*: for instance an inattentive human is more likely to miss feedbacks. Therefore we propose an attentional tunnelling fuzzy model for behavioural and physiological data aggregation based on expert knowledge. An experiment with robots and human operators has been carried out in order to tune and test this model. The results of the experiment tend to confirm the soundness of the model.

The lack of reversibility of the internal state changes is a source of possible human-machine bad interactions and should be listed among the *vulnerabilities*. The last part of the work focuses on a reversibility scale and on reversibility check

in order to assess the degree of reversibility of the human operator's actions.

8.2 Further work

8.2.1 Assessment of the possibilistic model

The model based on possibility theory appears to be promising but there is a need to check its validity. First, experiments have to be conducted such as the one that was conducted in our flight simulator. The objective will be to test the efficiency of the model to assess conflict with automation. Situation awareness techniques (e.g.: subjective measures as the Situation Awareness Rating Technique - SART, or objective measures as the Situation Awareness Global Assessment Technique - SAGAT) could be considered to evaluate the diagnoses of the possibilitic model. If the tests prove the uncertainty model to be useful it could be later integrated as part of a real time conflict manager. This manager could perform conflict detection/identification and solving tasks. The solving part of the function could be performed thanks to:

- Cognitives countermeasures: a feedback (e.g. a change in the H/M interface) that is meant to help the human to better understand and solve the conflicts;
- Adaptive automation in order to modify the automation behaviour.

.

8.2.2 Integration of other measures for assessing the human "state"

A way to enrich the prediction of the possibilistic approach would be to integrate not only flight parameters but also eye tracking data. Indeed, the use of oculometric measures may provide additional data to infer the human operator's perception and understanding of the conflict. As revealed by our experiments, conflicts may induce attentional tunneling and make pilots neglect relevant pieces of information. It is more likely that conflicts also lead to other kinds of impaired attentional behaviours such as excessive saccadic activity. This could be investigated through the analysis of the ballistic behaviour of the eye and on finding relationships between the eye behaviour and the operator's attentional state using adaptive neuro fuzzy inference system [RDR$^+$]. One potential interest of this approach is that it allows to get rid of the use of areas of interest that requires expert knowledge of the user interface.

8.2.3 Other conflict sources

In the possibilistic model we consider the loss of a feedback as the sole conflict source. The model could be further enriched thanks to the introduction of other conflict sources, for instance feedback misunderstandings, or wrong estimates of human actions effects. They can be modelled thanks to expert knowledge and the most common cases (with relevant plausibility) may be taken into account. They can also be defined from the studied vulnerabilities. In fact this is already the case as the *inhibited behaviours* are used a reference for the "slip" definition. We could go further with e.g. *mode inconsistencies*: the effect of a human action is almost all the time the same except for some special cases.

Nevertheless computational issues must be evaluated: indeed each new possible conflict source that would be taken into account would make the number of situations to compute and to stock in memory increase.

More generally further studies and further experiments may also be performed with other vulnerabilities, which may be used to define new conflict patterns for the Petri Net analysis.

Bibliography

[AD91] G. D. Abowd and A. J. Dix, *Giving undo attention*, Interacting with Computers **4** (1991), 317–342.

[BBS12] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, *Using formal verification to evaluate human-automation interaction in safety critical systems, a review*, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans. (2012), 14–17.

[BDP97] S. Benferhat, D. Dubois, and H. Prade, *Possibility theory*, Artificial Intelligence **92** (1997), 259–276.

[BFR52] H. Bahrick, P. Fitts, and R. Rankin, *Effect of incentives upon reactions to peripheral stimuli*, Journal of Experimental Psychology **44** (1952), no. 6, 400–406.

[BHJ99] D.B. Beringer and H.C. Harris Jr, *Automation in general aviation: Two studies of pilot responses to autopilot malfunctions*, The International Journal of Aviation Psychology **9** (1999), no. 2, 155–174.

[Bil96] E. Billings, *Aviation automation : the search for a human-centered approach*, Lawrence Erlbaum associates, Inc., Mahwah, NJ, USA, 1996.

[BL02] J. Bredereke and A. Lankenau, *A rigorous view of mode confusion*, Computer safety, reliability and security. 21st Int. Conf., Safecomp 2002, LNCS 2434, Springer, 2002, pp. 19–31.

[BM11] A. Biletzki and A. Matar, *Ludwig Wittgenstein*, The Stanford Encyclopedia of Philosophy (Edward N. Zalta, ed.), summer 2011 ed., 2011.

[BMPC98] R.W. Butler, S.P. Miller, J.N. Potts, and V.A. Carreno, *A formal methods approach to the analysis of mode confusion.*, 17th Digital Avionics Systems Conference., 1998, pp. C41/1–C41/8.

[Boa10] National Transportation Safety Board, *Loss of thrust in both engines after encountering a flock of birds and subsequent ditching on the hudson river.*, Aircraft Accident Report NTSB/AAR-10 /03 (2010).

[BR69] L. F. Bishop and P. Reichert, *The interrelationship between anxiety and arrhythmias*, Annual Meeting of the Academy of Psychosomatic Medicine (1969).

[Cas00] C. Castelfranchi, *Conflict ontology*, Computational conflicts, Springer, 2000, pp. 21–40.

[CBD02] L. Cowen, L. Ball, and J. Delin, *An eyemovement analysis of webpage usability*, Computers XVI: Memorable Yet Invisible, Proceedings of HCI'02 (London, UK), 2002.

[CCH08] J. Creissac Campos and M. D. Harrison, *Systematic analysis of control panel interfaces using formal tools*, Interactive Systems. Design, Specification, and Verification, Springer, 2008, pp. 72–85.

[CF00] C. Castelfranchi and R. Falcone, *Conflicts within and for collaboration*, Conflicting agents (C. Tessier, L. Chaudron, and H.-J. Müller, eds.), Springer, 2000, pp. 33–61.

[CGPF11] S. Combéfis, D. Giannakopoulou, Ch. Pecheur, and M. Feary, *A formal framework for design and analysis of human-machine interaction*, Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on, 2011, pp. 1801–1808.

[CJR00] J. Crow, D. Javaux, and J. Rushby, *Models and mechanized methods that integrate human factors into automation design.*, International Conference on Human-Computer Interaction in Aeronautics : HCI-Aero, 2000, pp. 163–168.

[CL90] P.R. Cohen and H.J. Levesque, *Intention is choice with commitment*, Artif. Intell. **42** (1990), no. 2-3, 213–261.

[Cox92] E. Cox, *Fuzzy fundamentals*, IEEE Spectrum **29** (1992), no. 19, 58–61.

[CSDP10] M. Causse, J.M. Sénard, J.F. Démonet, and J. Pastor, *Monitoring cognitive and emotional processes through pupil and cardiac response during dynamic versus logical task*, Applied psychophysiology and biofeedback **35** (2010), no. 2, 115–123.

[DCT11] F. Dehais, M. Causse, and S. Tremblay, *Mitigation of conflicts with automation*, Human Factors **53** (2011), no. 3, 448–460.

[Deh12] F. Dehais, *Conflits et persistance dans l'erreur: une approche pluridisciplinaire.*, Habilitation à diriger les recherches (2012).

[DF05] D. Dubois and Ph. Fortemps, *Selecting preferred solutions in the minimax approach to dynamic programming problems under flexible constraints*, European Journal of Operational Research **160** (2005), no. 3, 582–598.

[DP90] D. Dubois and H. Prade, *An introduction to possibilistic and fuzzy logics.*, Readings in uncertain reasoning (1990), 742–761.

[DP00] F. Dehais and P. Pasquier, *Approche générique du conflit.*, Ergonomie et Interaction Homme-Machine (ErgoIHM 2000) (2000), 56–63.

[DP07] D. Dubois and H. Prade, *Possibility theory*, Scholarpedia **2** (2007), no. 10, 2074.

[DP09] ———, *Formal representations of uncertainty*, Decision-Making Process (D. Bouyssou, D. Dubois, M. Pirlot, and H. Prade, eds.), Wiley, Hoboken, N.J. USA, 2009.

[DTC03] F. Dehais, C. Tessier, and L. Chaudron, *Ghost: Experimenting conflicts countermeasures in the pilot's activity*, International Joint Conference on Artificial Intelligence (IJCAI) (2003), no. 18, 163–168.

[DTCR09a] F. Dehais, C. Tessier, L. Christophe, and F. Reuzeau, *The perseveration syndrome in the pilot's activity: guidelines and cognitive countermeasures*, 7th International Working Conference on Human Error, Safety, and System Development HESSD (2009).

[DTCR09b] F. Dehais, C. Tessier, L. Christophe, and F. Reuzeau, *The perseveration syndrome in the pilot's activity: guidelines and cognitive countermeasures*, 7th International Working Conference on Human Error, Safety, and System Development (HESSD) (Brussels, Belgium), 2009.

[Dub86] D. Dubois, *Belief structures, possibility theory and decomposable measures on finite sets*, Computers and AI **5** (1986), 403–416.

[dVP11] E. de Visser and R. Parasuraman, *Adaptive aiding of human-robot teaming effects of imperfect automation on performance, trust, and workload*, Journal of Cognitive Engineering and Decision Making **5** (2011), no. 2, 209–231.

[DWM07] S.R. Dixon, C.D. Wickens, and J.S. McCarley, *On the independence of compliance and reliance: Are automation false alarms worse than misses?*, Human Factors: The Journal of the Human Factors and Ergonomics Society **49** (2007), no. 4, 564–572.

[Eas59] J. Easterbrook, *The effect of emotion on cue utilization and the organization of behavior*, Psychological review **66** (1959), no. 3, 183–201.

[End95] M. R. Endsley, *Towards a theory of situation awareness in dynamic systems*, Human Factors **1** (1995), no. 37, 32–64.

[Fea05] M. Feary, *Formal identification of automation surprise vulnerabilities in design*.

[Fea07] ———, *Automatic detection of interaction vulnerabilities in an executable specification*, D. Harris (Ed.): Engin. Psychol. and Cog. Ergonomics, HCII 2007, LNAI 4562 (2007), 487–496.

[Fea10] ———, *A toolset for supporting iterative human automation interaction in design*, NASA Ames Research Center, Tech. Rep. 20100012861 (2010).

[GTC05] J. Gow, H.W. Thimbleby, and P.A. Cairns, *Automatic critiques of interface modes*, DSV-IS, 2005, pp. 201–212.

[Han00] M. Hannebauer, *Their problems are my problems*, Conflicting agents - Conflict management in multi-agent systems (C. Tessier, L. Chaudron, and H.-J. Mueller, eds.), Kluwer Academic Publishers, 2000.

[Hol03] Orasanu J. McCoy C. Holbrook, J., *Weather-related decision making by aviators in alaska*, 2003, pp. 576–581.

[Ina03] T. Inagaki, *Automation and the cost of authority*, International Journal of Industrial Ergonomics **31** (2003), no. 3, 169–174.

[Jav02] D. Javaux, *A method for predicting errors when interacting with finite state systems. How implicit learning shapes the user's knowledge of a system*, Reliability Engineering and System Safety (2002), no. 75, 147–165.

[Jen95] N.R. Jennings, *Controlling cooperative problem solving in industrial multi-agent systems using joint intentions*, Artifificial Intelligence **75** (1995), no. 2, 195–240.

[JMH03] A. Joshi, S. P Miller, and M.P. Heimdahl, *Mode confusion analysis of a flight guidance system using formal methods*, Digital Avionics Systems Conference, 2003. DASC'03. The 22nd, vol. 1, IEEE, 2003, pp. 2–D.

[Kra97] S. Kraus, *Negotiation and cooperation in multi-agent environments*, Artifificial Intelligence **94** (1997), no. 1-2, 79–97.

[LC99] G. Lüttgen and V. Carreño, *Analyzing mode confusion via model checking*, SPIN, 1999, pp. 120–135.

[Los89] R.M.Jr. Losee, *Minimizing information overload: The ranking of electronic messages*, Journal of Information Science **15** (1989), no. 3, 179–189.

[LP97] N.G. Leveson and E. Palmer, *Designing automation to reduce operator errors*, 1997 IEEE International Conference on Systems, Man and Cybernetics (Orlando, FL), vol. 2, 1997, ISBN: 1-60566-256-9, pp. 1144–1150.

[LPS+97] N.G. Leveson, L.D. Pinnel, S.D. Sandys, S. Koga, and J.D. Reese, *Analyzing software specifications for mode confusion potential*, Workshop on Human Error and System Development (1997), 132–146.

[LSMC10] C. Layton, P. Smith, and C. Mc Coy, *Design of a cooperative problem-solving system for en-route flight planning: An empirical evaluation*, Human factors: The Journal of the Human Factors and Ergonomics Society **1** (2010), no. 36, 94–119.

[LT05] Ch. Lesire and C. Tessier, *Particle Petri nets for aircraft procedure monitoring under uncertainty.*, Proceedings of the Applications and Theory of Petri Nets conference, 2005, pp. 329–348.

[MA07] R.L. Mandryk and M.S. Atkins, *A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies*, International Journal of Human-Computer Studies (2007), no. 65, 329–347.

[MC99] M. Modarresa and S. W. Cheon, *Function-centered modeling of engineering systems using the goal tree-success tree technique and functional primitives*, Reliability Engineering and System Safety, vol. 64, 1999, pp. 181–200.

[MD00] H.J. Müller and R. Dieng, *Computational conflicts: Conflict modeling for distributed intelligent systems with contributions by numerous experts*, Springer, 2000.

[Mer11] S. Mercier, *Contrôle du partage de l'autorité dans un système d'agents hétérogènes*, Ph.D. thesis, ISAE, 2011.

[Mey01] J. Meyer, *Effects of warning validity and proximity on responses to warnings*, Human Factors **43** (2001), no. 4, 563–572.

[MM87] D. Marca and C. McGowan, *Structured analysis and design technique*, McGraw-Hill, 1987.

[MP05] U. Metzger and R. Parasuraman, *Automation in future air traffic management: Effects of decision aid reliability on controller performance and mental workload*, Human Factors: The Journal of the Human Factors and Ergonomics Society **47** (2005), no. 1, 35–49.

[MTD10a] S. Mercier, C. Tessier, and F. Dehais, *Détection et résolution de conflits d'autorité dans un système homme-robot*, Revue d'Intelligence Artificielle, numéro spécial "Droits et Devoirs d'Agents Autonomes" **24** (2010), no. 3/2010, 325–356.

[MTD10b] S. Mercier, C. Tessier, and Fr. Dehais, *Authority management in human-robot systems*, IFAC/IFIP/IFORS/IE Symposium on Analysis, Design, and Evaluation of Human-Machine Systems (Valenciennes, France), 2010.

[Myl04] J. Mylopoulos, *Conceptual modelling III. Structured analysis and design technique (SADT)*, http://www.cs.toronto.edu/ jm/2507S/Notes04/SADT.pdf, 2004.

[O'B95] D. P. O'Brien, *Human reasoning includes a mental logic*, Behavioral and brain sciences **32** (1995), no. 1, 96–97.

[OC07] M. Oaksford and N. Chater, *Bayesian rationality*, Oxford University Press, Oxford, 2007.

[Ora01] Martin L. Davison J. Orasanu, J., *Factors in aviation accidents: Decision errors.*, 209–225.

[OW90] C. M. Ozveren and A. S. Willsky, *Observability of discrete event dynamic systems*, IEEE transactions on automatic control **34** (1990), no. 7, 797–806.

[Pal95] E. Palmer, *"Oops, it didn't arm." A case study of two automation surprises*, Proceedings of the Eighth International Symposium on Aviation Psychology (Columbus, OH, USA) (R. S. Jensen and L. A. Rakovan, eds.), 1995, pp. 227–232.

[PHB02] R. Parasuraman, J. Hansman, and S. Bussolari, *Framework for evaluation of human-system-issues with ASDE-X and related surface safety systems*, AAR-100 Report, Federal Aviation Administration (Washington, DC., USA), 2002.

[PSD12] C. Pizziol, Tessier, S., and Fr. Dehais, *What the heck is it doing? better understanding human-machine conflicts through models*, Proceedings of the ECAI Workshop on Rights and Duties of Autonomous Agents (RDA2) 2012 (Montpellier, France), 2012.

[PW08] R. Parasuraman and C. Wickens, *Humans : Still vital after all these years of automation*, Human factors **50** (2008), no. 3, 511–520.

[RAC+12] F. Rome, G. Adam, J. Condette, M. Causse, and F. Dehais, *Go-around manoeuver: a simulation study.*, European Association for Aviation Psychology Conference (2012).

[Ras83] J. Rasmussen, *Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models*, IEEE Transactions on SMC **3** (1983), no. 13, 257–266.

[RCP99] J. Rushby, J. Crow, and E. Palmer, *An automated method to detect potential mode confusions*, Presentation slides.

[RDR+] N. Regis, F. Dehais, E. Rachelson, Ch. Thooris, S. Pizziol, M. Causse, and C. Tessier, *Formal detection of attentional tunneling in human operator-automation interactions*, Under review: IEEE Transactions on Human-Machine Systems.

[Rea90] J. Reason, *Human error*, Cambridge University Press (New York, NY), 1990.

[Ric09] S. Rice, *Examining single-and multiple-process theories of trust in automation*, The Journal of General Psychology **136** (2009), no. 3, 303–322.

[Rus02] J. Rushby, *Using model checking to help discover mode confusions and other automation surprise*, Reliability Engineering and System Safety, vol. 75, 2002, pp. 167–177.

[SA84] J. Sandson and M.L. Albert, *Varieties of perseveration*, Neuropsychologia **22** (1984), no. 6, 715–732.

[She11] T.B. Sheridan, *Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: Distinctions and modes of adaptation*, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on **41** (2011), no. 4, 662–667.

[SM06] N. Su and J. Mylopoulos, *Conceptualizing the co-evolution of organizations and information systems*, Conceptual Modeling - ER 2006 (Tucson, AZ, USA), 2006.

[SMW07] N. Sarter, R. Mumaw, and C. Wickens, *Pilots' monitoring strategies and performance on automated flight decks: An empirical study combining behavioral and eye-tracking data.*, Human factors. (2007), no. 49, 347–357.

[SW95] N.B. Sarter and D.D. Woods, *How in the world did we ever get into that mode? Mode error and awareness in supervisory control*, Human Factors: The Journal of the Human Factors and Ergonomics Society **37** (1995), no. 1, 5–19.

[SWB97] N.B. Sarter, D.D. Woods, and C.E. Billings, *Automation surprises*, Handbook of Human Factors and Ergonomics (2nd edition) (G. Salvendy, ed.), New York, NY: Wiley, 1997, pp. 1926–1943.

[TMFC00] C. Tessier, H.-J. Mueller, H. Fiorino, and L. Chaudron, *Agents' conflicts: new issues*, Conflicting agents - Conflict management in multi-agent systems (C. Tessier, L. Chaudron, and H.-J. Mueller, eds.), Kluwer Academic Publishers, 2000.

[TVS+07] Y. Tsai, E. Viirre, C. Strychacz, B. Chase, and T. Jung, *Task performance and eye activity: predicting behavior relating to cognitive workload*, Aviation, Space, and Environmental Medicine (2007), no. 78, 176–185.

[TW04] L. Thomas and C. Wickens, *Eye-tracking and individual differences in off-normal event detection when flying with a synthetic vision system display*, 48th Conference of the Human Factors and Ergonomics Society (Santa Monica, CA, USA), 2004, pp. 223–227.

[Vak00] S. Vakil, *Analysis of complexity evolution management and human performance issues in commercial aircraft automation systems*, Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.

[VGdVKT06] H. Van Ginkel, M. de Vries, J. Koeners, and E. Theunissen, *Flexible authority allocation in unmanned aerial vehicles*, Conference Proceedings of the Human Factors and Ergonomics Society (San Francisco, CA, USA), 2006.

[VJ02] S.S. Vakil and Hansman J.R.Jr., *Approaches to mitigating complexity-driven issues in commercial autoflight systems*, Reliability Engineering and System Safety, vol. 75, 2002, pp. 133–145.

[WA09] C.D. Wickens and A.L. Alexander, *Attentional tunneling and task management in synthetic vision displays*, International Journal of Aviation Psychology **19** (2009), no. 2, 182–199.

[WD07] C.D. Wickens and S.R. Dixon, *The benefits of imperfect diagnostic automation: A synthesis of the literature*, Theoretical Issues in Ergonomics Science **8** (2007), no. 3, 201–212.

[WDGH05] C.D. Wickens, S. Dixon, J. Goh, and B. Hammer, *Pilot dependence on imperfect diagnostic automation in simulated UAV flights: an attentional visual scanning analysis*, 13th Annual International Symposium of Aviation Psychology (Oklahoma City, OK, USA), 2005.

[WE66] G. Weltman and G. Egstrom, *Perceptual narrowing in novice divers*, Human Factors: The Journal of the Human Factors and Ergonomics Society **8** (1966), no. 6, 499–506.

[Wic05] C.D. Wickens, *Attentional tunneling and task management*, 13th International Symposium on Aviation Psychology (Dayton, OH), 2005.

[Wic08] _____, *Situation awareness: review of Mica Endsley's 1995 articles on situation awareness theory and measurement*, Human factors **50** (2008), no. 3, 397–403.

[Wil83] R. Wilensky, *Planning and understanding: A computational approach to human reasoning*, Reading, MA: Addison-Wesley., 1983.

[Wil85] L. Williams, *Tunnel vision induced by a foveal load manipulation*, Human Factors: The Journal of the Human Factors and Ergonomics Society **27** (1985), no. 2, 221–227.

[Wit53] L. Wittgenstein, *Philosophical investigations*, 1953.

[WS00] D. Woods and N. Sarter, *Learning from automation surprises and going sour accidents.*, Cognitive engineering in the aviation domain. (N. Sarter and R. Amalberti, eds.), 2000, p. 347.

Prédiction de conflits dans des systèmes homme-machine

Sergio Pizziol

Introduction

Le 31 mai 2009, à 22 h 29, l'Airbus A330 effectuant le vol AF 447 décolle de l'aérodrome de Rio de Janeiro Galeão à destination de Paris Charles de Gaulle.(...) Vers 2 h 00, le commandant de bord quitte le poste de pilotage. Vers 2 h 08, l'équipage effectue une déviation de 12 degrés vers la gauche, probablement pour éviter des échos détectés par le radar météo.
A 2 h 10 min 05, vraisemblablement à la suite de l'obstruction des sondes Pitot par des cristaux de glace, les indications de vitesse deviennent erronées et des automatismes se désengagent. La trajectoire de l'avion n'est pas maîtrisée par les deux copilotes. Ils sont rejoints 1 minute 30 plus tard par le commandant de bord, alors que l'avion est dans une situation de décrochage qui se prolonge jusqu'à la collision avec la mer, à 2 h 14 min 28.
L'accident résulte de la succession des événements suivants :

- *l'incohérence temporaire entre les vitesses mesurées, vraisemblablement à la suite de l'obstruction des sondes Pitot par des cristaux de glace ayant entraîné notamment la déconnexion du pilote automatique et le passage en loi alternate ;*
- *les actions inappropriées sur les commandes déstabilisant la trajectoire ;*
- *l'absence de lien, de la part de l'équipage, entre la perte des vitesses annoncée et la procédure adaptée ;*
- *l'identification tardive par le PNF de l'écart de trajectoire et la correction insuffisante par le PF ;*
- *la non identification par l'équipage de l'approche du décrochage, l'absence de réaction immédiate et la sortie du domaine de vol ;*
- *l'absence de diagnostic de la part de l'équipage de la situation de décrochage et en conséquence l'absence d'actions permettant de la récupérer.*

Le BEA a adressé 41 recommandations de sécurité à la DGAC, à l'EASA, à la FAA, l'OACI et aux autorités brésiliennes et sénégalaises qui portent sur les enregistreurs de vol, la certification, la formation et l'entraînement des pilotes, la suppléance du commandant de bord, le SAR et l'ATC, les simulateurs, l'ergonomie du poste de pilotage, le retour d'expérience opérationnel et la surveillance des exploitants français par l'Autorité nationale de surveillance.

Il s'agit d'un extrait du synopsis du rapport final du BEA (Bureau d'Enquêtes et d'Analyses) sur l'accident du vol AF 447 Rio de Janeiro - Paris . Dans la conclusion de ce même document, parmi les *faits établis par l'enquête* :

- *en l'absence de la présentation des vitesses limites sur le bandeau de vitesse du PFD, l'alarme de décrochage sonore n'est confirmée par aucune indication visuelle spécifique*
- *l'alarme de décrochage a retenti de façon continue pendant 54 secondes*
- *aucun des pilotes n'a formellement identifié la situation de décrochage*
- *l'incidence de l'avion n'est pas directement présentée aux pilotes*

Et parmi les causes de l'accident sont mentionnés :

- *la non-identification par l'équipage de l'approche du décrochage, l'absence de réaction immédiate et la sortie du domaine de vol*
- *l'absence de diagnostic de la part de l'équipage de la situation de décrochage et en conséquence l'absence d'actions permettant de la récupérer*
- *un travail en équipage affaibli par l'incompréhension de la situation à la déconnexion du PA (...)*
- *l'absence d'indication claire dans le poste de pilotage de l'incohérence des vitesses identifiée par les calculateurs*

Donc parmi les causes de l'accident, le BEA a identifié la mauvaise évaluation de la situation de la part de l'équipage, en partie en raison des incohérences dans les informations affichées, des alertes qui n'ont pas été perçus et de l'absence d'informations pertinentes sur les écrans.

En 2002, une étude du Massachusetts Institute of Technology a rapporté 184 incidents attribués à une mauvaise conscience de situation dans le *NASA Aviation Safety Reporting Systems*.

C'est pourquoi le développement de méthodes et de systèmes qui peuvent détecter et remédier à ces situations est essentielle. Le travail qui a été effectué au cours de cette thèse s'inscrit dans le cadre de la recherche consacrée aux problèmes liés à une mauvaise conscience de situation, aux surprises d'automatisation et aux conflits entre l'humain et la machine qui pourraient découler de ces situations. Plus précisément cette thèse concerne la modélisation, l'analyse et la prédiction de conflit homme-machine dans des systèmes critiques et plus particulièrement les systèmes de pilotage et de contrôle d'engins autonomes.

Contexte

Nous proposons une structure générique de système humain-machine (voir figure 5), sur laquelle s'appuient les travaux de thèse. La définition et description des

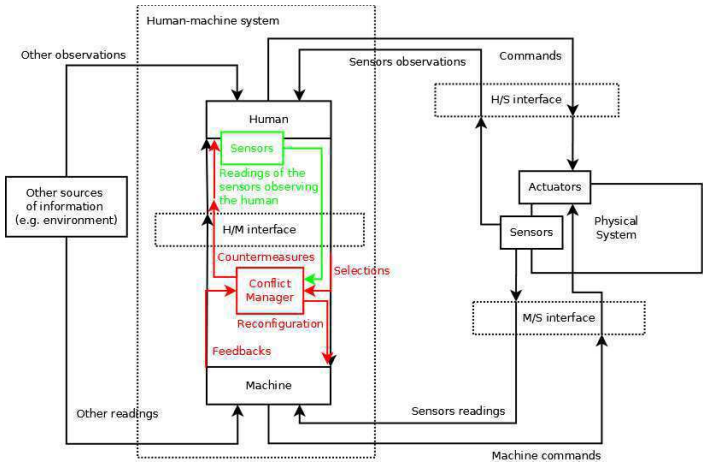


FIGURE 1 – Système homme-machine avec gestion des conflits

principes du partage d'autorité sur une ressource est aussi nécessaire (voir table 1) afin de pouvoir gérer le partage des ressource non partageables entre l'homme et la machine, qui peut constituer une source de conflit.

Agent	Authority	Access	Preemption	Interruptions	Description	Symmetric relation
X Y	No access Preemption	No Yes	- Yes	- No	Not Sharing	No
X Y	Access Preemption	Yes Yes	No Yes	Yes No	Exclusionary Sharing	No
X Y	Access Access	Yes Yes	No No	No No	Cooperative Sharing	Yes
X Y	Preemption Preemption	Yes Yes	Yes Yes	Yes Yes	Preemptive Sharing	Yes

TABLE 1 – Relation d'autorité entre deux agents.

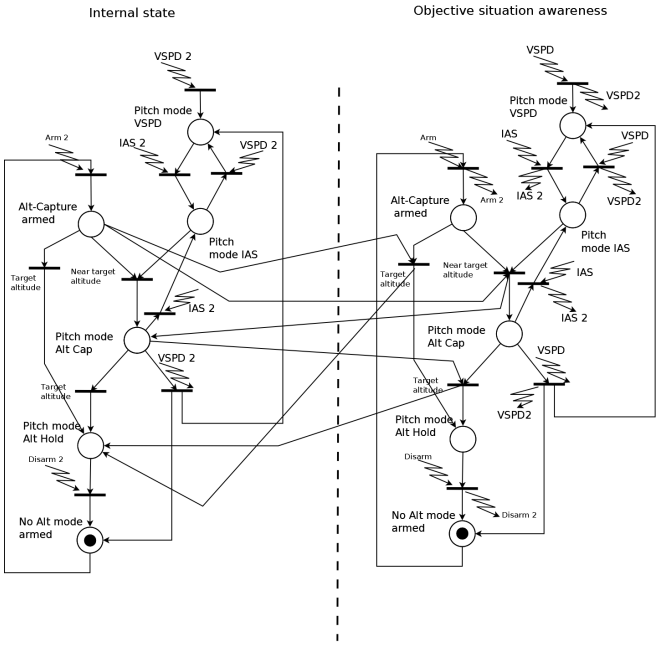
Conflits

Certaines définitions du conflit données dans la littérature sont focalisées su un point de vue *logique*, et expriment les conflit sous la forme d'inconsistances entre buts. D'autre auteurs modélisent les conflits (dits *physiques*), comme des incohérences issues d'accès concurrentiels à des ressource uniques et non partageables. Dans d'autres cas les conflits sont issus des incohérences entre les *connaissances* des agents.

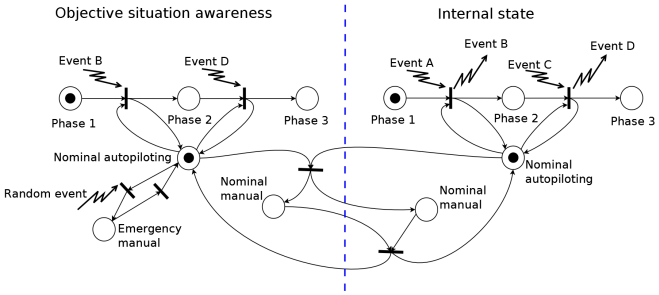
De manière générale, ces travaux se focalisent sur un cas spécifique et essaient ensuite de généraliser la définition. D'autres travaux proposent des définitions générales qui néanmoins implicitement gardent la séparation entre les trois cas. En comparant ces définitions en regard de ces différentes acceptions du terme conflit nous sommes arrivé à la conclusion que les généralisations ne sont pas utiles à la résolutions des conflits, parce qu'ils sont de nature fondamentalement différente.
Nous proposons une nouvelle définition conceptuelle qui ne masque pas la différence entre les cas. Cette distinction permet de focaliser l'étude sur les conflits portant sur les *connaissances*.
Deux notions clés sont importantes pour la prédiction des conflits : la propriété de *contrôle complet* (qui sous-tend à tout moment une compréhension correcte de l'état du système par l'opérateur) et les *vulnérabilités* (qui correspondent à des changements d'états internes du système potentiellement porteurs de conflits à venir). Ces deux aspects sont par la suite au cœur de la modélisation proposée.

Patterns du conflit en réseau de Petri

Nous avons développé une méthode de prédiction *a priori* de conflits basée sur un double réseau de Petri. La méthode est formulé suite à l'observation de deux cas réels (voir figure 2) .



(a) Un senario de *kill the capture*



(b) Le senario dit *rain and automation*

FIGURE 2 – Les deux cas réels.

Nous formulons une version plus générique du modèle pour pouvoir modéliser les conflits standard (voir figure 3), les problèmes liés aux limites d'autorité de

l'opérateur ou encore des conflits liés à des effets de bords. Plus en détaille la méthode se base sur la recherche de transitions bloquantes dans le double réseau de Petri, transitions qui correspondent à changements d'états internes qui si non détectés devraient amener à un conflit.

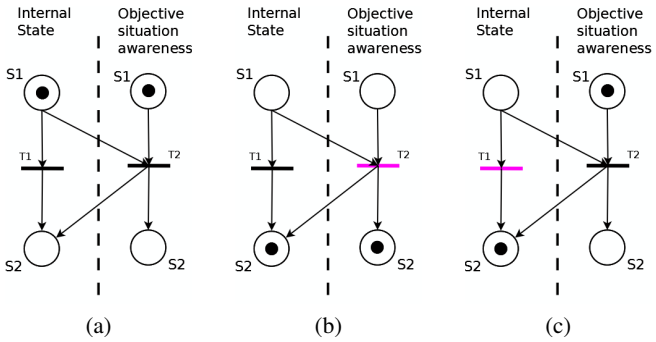


FIGURE 3 – Un changement automatique.

Une expérience en simulateur de vol

Afin de tester la validité du modèle générique nous avons conçu une expérience : un cas concret de manipulation d'un autopilote simplifié mis en œuvre dans trois scénarios de navigation aérienne (voir figure 4).

FIGURE 4 – Simulateur de vol.

Les scénarios d'expérimentation ont été pensés pour tester les comportement des pilotes et vérifier si le transitions d'état identifiés comme critiques (grâce au modèle générale) peuvent réellement amener à des conflit. En d'autre termes l'hypothèse principale de l'expérimentation vise à établir que des changements d'état internes du système non (ou mal) représentés sur l'interface qui amènent à un blocage du réseau de Petri aboutissent dans quelque cas à des conflits homme-machine. L'expérience montre que en effet les situation identifiés comme critiques

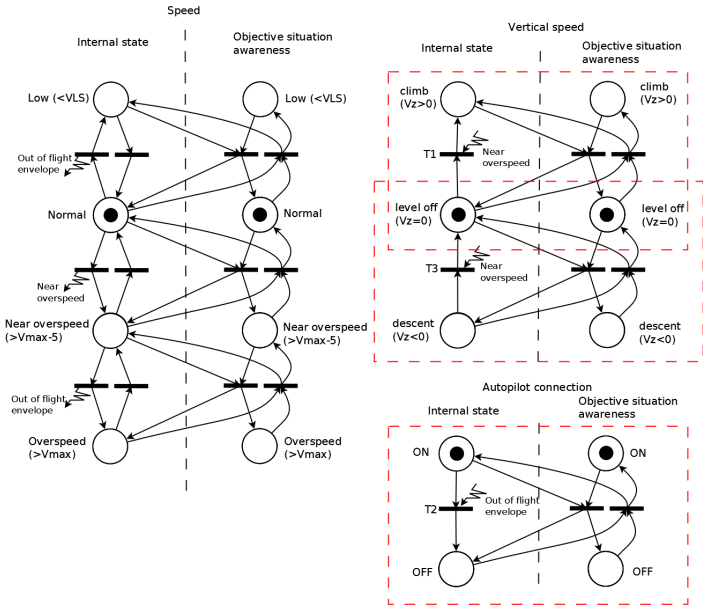


FIGURE 5 – Le modèle en réseau de Petri de la logique du pilote automatique.

peuvent amener à des cas de conflit (voir tables 2,3,4).

Participants	10
Conflicts cases	**9**
Conflict detection	2
Conflict solved	**1**

TABLE 2 – Premier scénario de conflit : réversion de mode *near overspeed*.

Participants	10
Conflicts cases	**10**
Conflict detection	10
Conflict solved	**7**

TABLE 3 – Deuxième scénario de conflit : déconnexion automatique du pilote automatique.

Il faut remarquer que la méthode ne peut pas prédire en temps réel l'apparition des conflits car le modèle repose sur une hypothèse pessimiste : l'opérateur

Participants	10
Conflicts cases	**9**
Conflict detection	8
Conflict solved	**1**

TABLE 4 – Troisième scénario de conflit : réversion de mode *near overspeed* avec *altitude sélecté incohérente*.

n'apercevra pas les changements d'état mal affichés. Cette méthode est néanmoins utile pour identifier dès la conception des situation critiques qui pourraient amener à des conflits. Relaxer cette hypothèse ouvre la porte pour le chapitre suivant.

Une estimation du suivi de situation humain de l'état interne

L'hypothèse du chapitre précèdent est remplacée par une autre qui laisse la place à l'incertain : normalement l'opérateur aura aperçu tout changement d'état, sauf si ses action en font douter. Il faut remarque que cette méthode pour la gestion des incertitudes permettra aussi de formuler des hypothèse pessimistes mais toujours avec un degré d'incertitude. De plus la modélisation des erreur d'interprétation autre que la simple omission est aussi bien possible.
Plus dans le détail ce chapitre propose un modèle possibiliste permettant d'évaluer la plausibilité de plusieurs état et par la suite de sélectionner le plus plausible parmi les états candidats. Cette démarche est en effet nécessaire en raison du caractère multivoque de la relation entre états internes réels et états internes interprétés (voir figure 6). La démarche s'appuie sur l'écriture d'une table logique contenant la to-

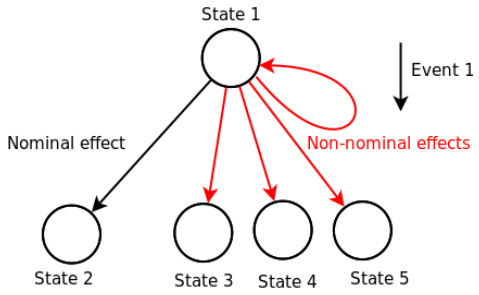


FIGURE 6 – Représentation des plusieurs effets possibles sur les internes interprétés et un changement d'états internes réels.

talité des comportement automatisés et déclenches par une action humaine, les changements résultants dans l'état interne, et les possibles erreurs d'interprétation de l'affichage (supposé correct).

L'ensemble de cette combinatoire est ensuite traité à partir d'équations qui régissent les possibilités respectives d'erreurs, de problème d'attention, ou d'exécution correcte de séquences de commandes, permettant en finale de distinguer la configuration la plus possible. Ces équation sont *bloquées* et calibrés une fois pour toutes dans les exemples montrés. Il faut remarquer que il s'agit dans les deux cas d'applications démonstratives de l'idée. En principe elles pourraient être function de l'expérience et de l'age de l'opérateur, de son état de fatigue et son état attentionnel (pour une évaluation de l'état attentionnel voir prochain chapitre). Ces règles édictées pour *classer* les possibilités ne sont pas forcement les plus appropriés :

		c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14
SITUATION															
Selection	Up	1	1	1											
	Down				1	1	1								
Event	High to Medium							1				1			
	Medium to Low								1				1		
	Low to Medium									1				1	
	Medium to High										1				1
State															
Variable	Low	1			1					1					
	Medium		1			1			1		1				
	High			1			1	1							
BEHAVIOUR															
State															
Variable	Low				1	1			1						
	Medium	1					1	1		1					
	High		1	1							1				
POSSIBILISTIC															
	Nominal	1	1	1	1	1	1	1	1	1	1	0	0	0	0
	Non nominal version of											c7	c8	c9	c10
	Possibility	normal	normal	slip	slip	normal	normal	normal	normal	normal	normal	lost feedback	lost feedback	lost feedback	lost feedback
	Description	Selection Up, from Low to Medium	Selection Up, from Medium to High	Selection Up, no effect because already High	Selection Down, no effect because already Low	Selection Down, from Medium to Low	Selection Down, from High to Medium	Automated behaviour High to Medium	Automated behaviour Medium to Low	Automated behaviour Low to Medium	Automated behaviour Medium to High	Automated behaviour High to Medium, but unseen	Automated behaviour Medium to Low, but unseen	Automated behaviour Low to Medium, but unseen	Automated behaviour Medium to High, but unseen

TABLE 5 – Table logique pour un exemple jouet.

elle constituent un set arbitraire pour démontrer les potentialité de la méthode. Néanmoins les problèmes de combinatoire pour véritable système complexe restent à estimer.

Les résultat sur un cas réaliste de simulateur de vol sombrerait confirmer le caractère raisonnable de la méthode, sans pour autant en prouver la validité.

Modèle floue de la tunnelisation attentionnelle

Nous avons conduit une expérimentation destinée à identifier, à partir d'indicateurs physiologiques, des phénomènes des persévération provoquant un défaut d'attention des opérateurs vis-à-vis de signaux d'alarme émis par le système. La mise en œuvre d'un système d'agrégation flue permet d'aboutir à des résultat apparemment cohérents et prometteurs (voir figure 8), mais est pour l'instant *problème dépendants*. Une approche plus générale est développée au sein du même groupe de travail. Cette méthode ne dépendrait pas (ou peu) de l'interface.

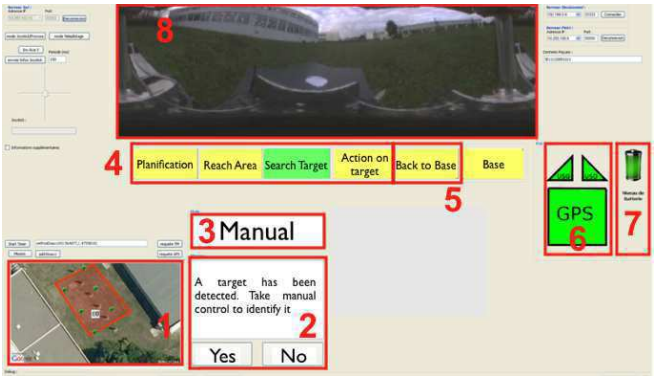
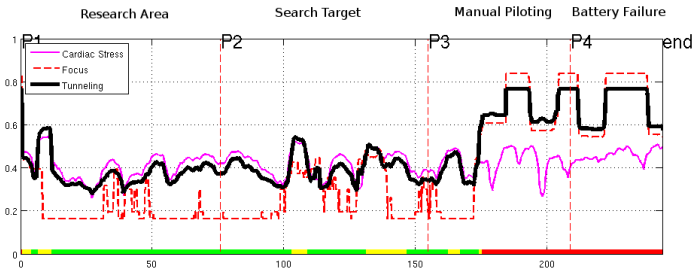
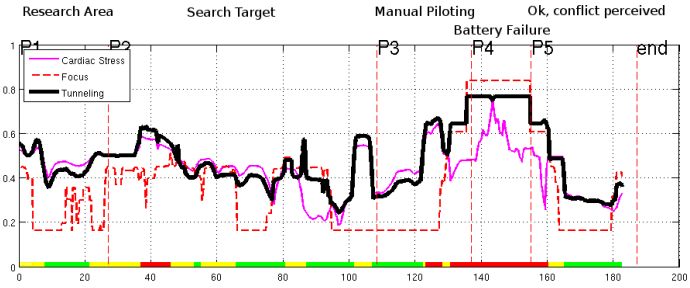


FIGURE 7 – L'interface homme/machine utilisée pour cette expérience.



(a) Cas A : tunneling attentionnel.



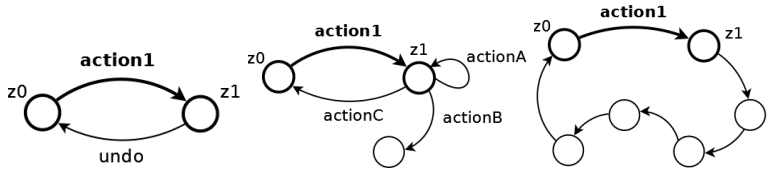
(b) Cas B : Ok, conflit aperçu.

FIGURE 8 – Les sorties du système d'agrégation flue pour deux participants.

Réversibilité

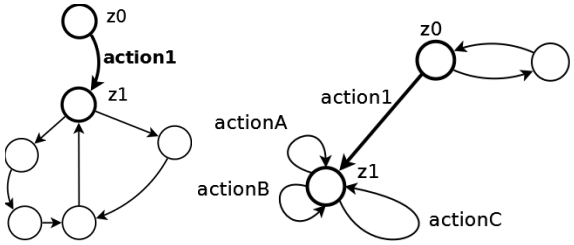
Nous avons traité le problème de la réversibilité des états d'un système et la façon de le détecter à partir d'une table logique. Nous définissions plusieurs propriétés de réversibilité (voir figures 9, 10). Ces propriétés sont ensuite positionnés

sur une échelle de réversibilité (voir table 6).



(a) Undo. (b) Réversibilité en une seule action. (c) Réversibilité.

FIGURE 9 – Propriétés de réversibilité.



(a) Changement d'état irréversible. (b) Changement d'état totalement irrécupérable.

FIGURE 10 – Propriétés de irréversibilité.

Name	Property
Undo	A particular action to come back to z0
One action reversible	An action to come back to z0
Reversible	A sequence of actions to come back to z0
Irreversible	No sequence of actions to come back to z0
Eventually totally unrecoverable	Any sequence of actions leads to a blocked state
Totally unrecoverable	No action to leave z1

TABLE 6 – Échelle de réversibilité.

Nous avons développé une méthode qui s'appuie sur l'approche ADEPT (Automation Design and Evaluation Prototyping Toolset) qu'elle enrichit de la vérification de certaines propriétés de réversibilité. La méthode a été démontrée sur deux exemples.

Perspectives

Évaluation du modèle possibiliste

Le modèle basé sur la théorie possibliste est prometteur mais nécessite d'une vérification. En premier lieu des autres expériences au simulateur de vol doivent être conduites. L'objectif sera de tester l'efficacité avec la quelle la méthode détecte des conflit homme-machine. Si le modèle possibiliste s'avère être utile il pourrait ensuite être intégré dans un système de gestion des conflit en temps réel. Ce système devrait garantir des fonction de détection/identification an résolution de conflits. La résolution pourrait être obtenue résolution grâce à :

- Contre-mesures cognitives : un feed-back conçu pour aider l'humain à mieux comprendre la situation et à résoudre le conflit
- Automation adaptative pour modifier le comportement de la machine.

Intégration d'autre mesure pour l'estimation de l'"état" de l'opérateur

Une façon d'enrichir la prédiction de l'approche possibiliste serait d'intégrer non seulement les paramètres de vol mais aussi les données de l'eye tracker. En effet, l'utilisation des mesures oculométriques peut fournir des données supplémentaires pour déduire la perception de l'opérateur humain et sa compréhension du conflit. Comme révélé par nos expériences, des conflits peuvent induire tunneling attentionnel et faire en sorte que les pilotes négligent éléments d'information pertinents. Il est possible que les conflits conduisent également à d'autres types de comportements de déficit attentionnel comme une activité saccades excessive. Cela pourrait être étudiée grâce à l'analyse du comportement balistique de l'œil et à l'identification de relations entre le comportement de l'œil et l'état attentionnel de l'opérateur à l'aide d'un système neuro floue d'inférence. Un intérêt potentiel de cette approche est qu'elle permet de se débarrasser de l'utilisation des zones d'intérêt qui nécessite des connaissances d'expert de l'interface utilisateur.

Autres sources de conflit

Dans le modèle possibiliste nous considérons la perte d'un retour d'information comme la seul source de conflits. Le modèle pourrait être enrichi grâce à l'introduction d'autres sources de conflits, des malentendus des feedbacks, ou des estimations erronées des effets des actions humaines. Ils peuvent être modélisés grâce à des connaissances d'expert, les cas plus courants (avec leur dégrée de plausibilité) peuvent être pris en compte. Ils peuvent également être définis à partir des vulnérabilités étudiées. En fait, c'est déjà le cas pour comportements dits *inhibés*, qui servent de référence pour la définition des erreurs humaines. Nous pour-

rions aller plus loin avec par exemple les *incohérences de mode* : l'effet d'une action humaine est presque tout le temps le même, sauf pour certains cas particuliers. Néanmoins problèmes de combinatoire doivent être évalués : en effet chaque nouvelle possible source de conflit qui sera prise en compte rendrait de plus en plus important le nombre des cas à calculer et à stocker dans la mémoire. Plus généralement d'autres études et d'autres expériences peuvent également être effectuées avec d'autres vulnérabilités, qui peuvent être utilisés pour définir de nouveaux modèles de conflit pour l'analyse réseau de Petri.