Petri net-based modelling of human–automation conflicts in aviation

Sergio Pizziol[a]*, Catherine Tessier[a] and Frédéric Dehais[b]

[a]*Onera, Toulouse, France;* [b]*ISAE, Toulouse, France*

(*Received 2 October 2012; accepted 15 December 2013*)

Analyses of aviation safety reports reveal that human–machine conflicts induced by poor automation design are remarkable precursors of accidents. A review of different crew–automation conflicting scenarios shows that they have a common denominator: the autopilot behaviour interferes with the pilot's goal regarding the flight guidance *via* 'hidden' mode transitions. Considering both the human operator and the machine (i.e. the autopilot or the decision functions) as agents, we propose a Petri net model of those conflicting interactions, which allows them to be detected as deadlocks in the Petri net. In order to test our Petri net model, we designed an autoflight system that was formally analysed to detect conflicting situations. We identified three conflicting situations that were integrated in an experimental scenario in a flight simulator with 10 general aviation pilots. The results showed that the conflicts that we had a-priori identified as critical had impacted the pilots' performance. Indeed, the first conflict remained unnoticed by eight participants and led to a potential collision with another aircraft. The second conflict was detected by all the participants but three of them did not manage the situation correctly. The last conflict was also detected by all the participants but provoked typical automation surprise situation as only one declared that he had understood the autopilot behaviour. These behavioural results are discussed in terms of workload and number of fired 'hidden' transitions. Eventually, this study reveals that both formal and experimental approaches are complementary to identify and assess the criticality of human–automation conflicts.

Practitioner Summary: We propose a Petri net model of human–automation conflicts. An experiment was conducted with general aviation pilots performing a scenario involving three conflicting situations to test the soundness of our formal approach. This study reveals that both formal and experimental approaches are complementary to identify and assess the criticality conflicts.

Keywords: human–machine system; conflict solving; automation surprise; Petri net

1. Introduction

In aviation psychology, the analysis of air safety reports (Holbrook, Orasanu, and McCoy 2003) has shown that the occurrence of a cognitive conflict is a remarkable precursor to the degradation of human operators' performance, bringing about plan continuation errors (Orasanu et al. 2001). Experiments conducted in flight simulators have revealed that such conflicts could lead to behavioural patterns that indicate perseveration (Dehais, Tessier, and Chaudron 2003; Dehais et al. 2009). It is assumed that, once a human operator is trapped in perseveration behaviour, most of their attention resources are summoned up towards conflict solving to the detriment of primary flight parameters monitoring (e.g. alarms). Conflicts do occur not only between humans, but also between a human operator and artificial systems (Wiener and Curry 1980; Hoca, Youngb, and Blossevillec 2009), which is known as 'automation surprises' (Sarter and Woods 1995; Sarter, Woods, and Billings 1997). In aeronautics, such situations stem from the fact that the autopilot does not behave as expected by the crew. This cooperation breakdown can lead to accidents with an airworthy airplane when the crew persist in solving a minor conflict (Billings 1996) 'instead of switching to another means or a more direct means to accomplish their flight path management goals' (Woods and Sarter 2000, 347), and this can occur despite auditory alarms (Beringer and Harris 1999). Such hazardous situations can be explained in terms of careless design of authority sharing (Inagaki 2003; Dehais, Causse, and Tremblay 2011) and level of automation (Endsley 1999; Parasuraman 2000). Indeed, modern commercial planes can be considered controlled by a human agent (i.e. the pilot or the crew) and an artificial agent (the autopilot and the flight management system) that share a common resource, i.e. the flight guidance (Weyer 2006). As these two agents have different 'sensors', different 'logics' and different knowledge, and may take over this common resource from each other, conflicts are likely to occur. Moreover, the role and the authority of both agents are pre-defined and may fail to adapt in abnormal operation. From the human operator's point of view, these conflicts are the consequences of a lack of knowledge of the automation (Javaux 2002): the human operator may fail to behave accurately (i.e. take over or reconnect automation). Indeed, the automatic mode transitions are not always detected or understood by the operators, which may have catastrophic consequences (Sarter, Mumaw, and Wickens 2007). Such considerations should lead to provide guidelines to avoid poor authority sharing design and to anticipate on-board conflicts.

*Corresponding author. Email: sergio.pizziol@gmail.com

Formal studies have been carried out to identify crew–autopilot conflicts. Finite state automata (Lesire and Tessier 2005) are generally used for modelling the autopilot since its behaviour is known, discrete and deterministic (Crow, Javaux, and Rushby 2000; Javaux 2002). Some authors (Leveson and Palmer 1997; Butler et al. 1998) have used this approach to examine the design of an autopilot and its tolerance to human error. Rushby (2002) 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 (Rushby, Crow, and Palmer 1999) that this approach could describe and predict a particular conflict between an MD88 autopilot and the crew. 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 (Crow, Javaux, and Rushby 2000). Therefore, an alternative approach must be considered.

In the literature, poor crew–automation interactions designs have been described from accident analyses, and they can be classified in three categories:

- *automated behaviour* (Feary 2005) or 'indirect mode changes' (Sandys et al. 1997) or 'armed behaviour' (Feary 2005): the autopilot automatically changes states without any simultaneous pilot's action (e.g. the automatic disconnection of the autopilot without any pilot's action);
- *operator authority limits* (Sandys et al. 1997), 'inhibited behaviour' (Feary 2005): the pilot's actions have no effect on the system, i.e. the autopilot does not follow the pilot's orders;
- Unintended *side effects* (Sandys et al. 1997): a unique pilot's action leads to a cascade of state changes in the autopilot (e.g. an action on the lateral guidance mode has unexpected consequences on the vertical guidance mode).

Obviously, these situations do not systematically induce conflicts, but some authors (Sandys et al. 1997; Feary 2005; Creissac Campos and Harrison 2008) have demonstrated their vulnerabilities when combined with a lack of appropriate automation feedback on the user's interface through displays or alerts.

In the present study, we propose a generic formal approach based on Petri nets to model pilot–automation conflicts and identify bad interaction patterns, i.e. critical transitions the interaction designer should be aware of. These patterns are then used to identify potential pilot–automation conflicts in the Petri net model of an autopilot system that we designed from different transportation and light aircraft autopilots. Nevertheless, this formal approach cannot predict the effects of these potential conflicts on the pilot's behaviour. Therefore, experiments have to be conducted in order to assess the actual pilot's behaviour when facing the a-priori detected critical transitions. To this end, 10 general aviation pilots were placed in our flight simulator that was equipped with our autopilot, so as to perform a scenario involving the different potential conflicting situations. Their flight performance and post-experiment debriefing were used to assess whether these situations had actually led to conflicts with the automation, and whether they had been detected or not, and solved or not.

2. Towards generic models of human–automation conflicts

An analysis of the catalogue of the poor interactions (see Introduction) shows that they have a common denominator: the autopilot behaviour interferes with the pilot's goal regarding flight guidance. More technically, a transition from a flight guidance mode to another one (e.g. 'climb' mode to 'level-off' mode) is usually the result of a direct action of the crew on the autopilot user interface. However, the autopilot behaviour can also result from a system event (e.g. overspeed) that automatically triggers transitions.

Such automatic transitions may be notified to the pilot (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 pilot, the feedback is likely to be missed: the transition is 'unseen'.

Starting from different real cases of 'hidden' or 'unseen' transitions, we have modelled them with Petri nets (Pizziol, Tessier, and Dehais 2012) in order to assess their consequences: firing those automatic transitions in the Petri nets may lead to deadlocks and we have noticed from the real cases that the deadlocks correspond to conflicting situations.

The next sections present our generic formal method to identify these 'hidden and blocking transitions' in human–machine system models. A case study is then proposed to illustrate the formal approach.

2.1 Petri nets for representing interaction patterns

A Petri net (David and Alla 2005) $\langle P, T, F, B \rangle$ is a bipartite graph with two types of nodes: P is a finite set of places; and T is a finite set of transitions. Arcs are directed and represent the forward incidence function F: $P \times T \rightarrow \mathbb{N}$ and the backward incidence function B: $P \times T \rightarrow \mathbb{N}$, respectively. An interpreted Petri net is such that conditions and events are associated with places and transitions. When the conditions corresponding to some places are satisfied, tokens are assigned to those

places and the net is said to be marked. The evolution of tokens within the net follows transition firing rules, i.e. a transition is fired when its upstream places contain enough tokens and the event associated with the transition occurs; the result of transition firing is that its upstream places are demarked, whereas its downstream places are marked. Petri nets are well suited to represent sequencing, parallelism and synchronisation.

As previously mentioned, an autopilot mode transition can be triggered by the pilot's action or by an automatic system event. In our Petri net representation of the pilot–autopilot system behaviour, the autopilot state will be represented by a token and the pilot's 'objective situation awareness' by another token. The 'objective situation awareness' is inferred, thanks to the pilot's actions on the autopilot (we assume that they are aware of their actions) and thanks to the automated behaviours that are triggered by the autopilot. For the automated behaviours, two scenarios are possible. In the first case, a feedback to inform the pilot is actually sent and perceived by the pilot (i.e. the 'objective situation awareness' is updated). In the second case, the feedback is either missing or not perceived by the pilot (therefore the 'objective situation awareness' remains unchanged).

We assume that the pilot knows about the logic of the autopilot and the meaning of the feedbacks: if the pilot is aware of a state change from S1 to S2, they will actually believe that the resulting state of the system is S2.

Because the pilot can only act on the interface and the automation can always 'read' the actions on the interface, the pilot-triggered transitions will affect both the state of the autopilot and the pilot's objective situation awareness. For instance, let us consider a scenario in which the pilot acts on the autopilot to change its state. The states of the autopilot are successively S1 then S2. In Figure 1 (left), the state of the autopilot is S1 and both agents'' (pilot's and autopilot's) knowledge is the same. The result of the firing of transition T1 (state change) is that both agents'' knowledge about the autopilot state is S2 (right).

The next paragraph deals with less symmetrical cases corresponding to the crew–automation interaction poor designs described in the Introduction.

2.2 The automated behaviour pattern

Let us consider a scenario in which the pilot and the autopilot can both act the autopilot to change its state. The firing of transition T2 (i.e. pilot's action or 'perceived' automated transition) makes the autopilot evolve from state S1 to S2 (see Figure 2, left), and both the autopilot state and the pilot's objective situation awareness are the same. On the contrary (right), the firing of transition T1 only makes the autopilot state evolve to S2, whereas the pilot's objective situation awareness is still S1. The case of a lack of feedback for T1 (i.e. '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 are the same conflict: the pilot is not aware of the action performed by the autopilot.

Nevertheless, this is a matter of semantic inconsistency and not of formal inconsistency 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 inconsistency (i.e. the autopilot state is S2, whereas the pilot's objective situation awareness 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 relevant patterns may be derived as variants of this one: the key point is the fact that there is an automatic 'hidden' or 'unseen' transition from the pilot's side.

Two other pattern variants are presented hereafter in order to cover the other poor crew–automation interaction designs that are described in the Introduction.

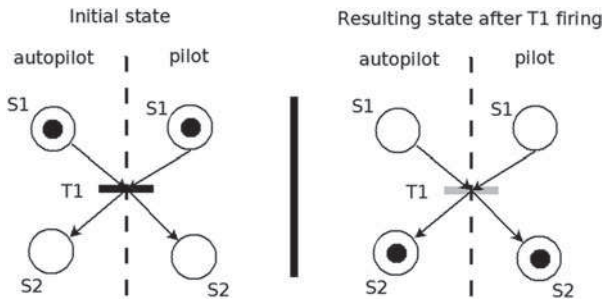

Figure 1. Shared transition: the autopilot state and the pilot's situation awareness are identical.

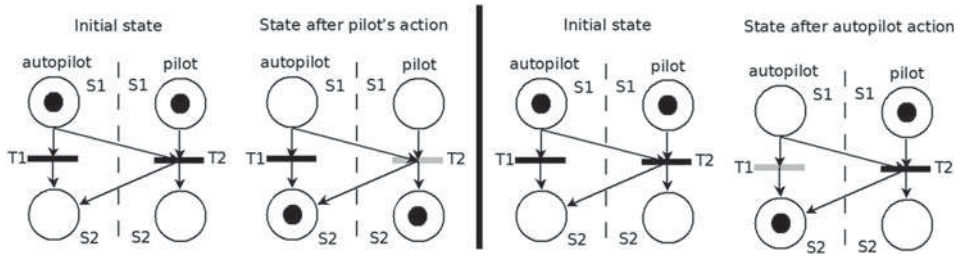

Figure 2. Automated behaviour.

2.2.1 *Pattern variant 1: operator authority limits*

The firing of transition T2 (pilot's action) makes the autopilot and pilot's objective situation awareness evolve to S2 (Figure 3 – middle). Nevertheless, the firing of the 'hidden' transition T1 (autopilot action) results in different states for the autopilot (S1) and the pilot's objective situation awareness (S2). Semantically, this is a conflict: *the pilot's action has been nullified by the autopilot, and the pilot is not aware of that.* Structurally, transition T2 is dead (Figure 3 – right).

2.2.2 *Pattern variant 2: side effect*

The firing of transition T1 makes the autopilot state and pilot's objective situation awareness evolve to S2 for variable A, and the autopilot state to S4 for variable B. The result is that the pilot's knowledge for variable B is S3, whereas the autopilot state is S4. Semantically, this is a conflict: *the pilot is not aware of a side effect of an action.* Structurally, transition T2 is dead (Figure 4 – bottom).

3. Formal identification of patterns in an autopilot model

The objective of this section 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).

3.1 Autopilot logic

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

- The autopilot is engaged and disengaged via a push button 'AP' on the flight control unit (FCU). 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 FCU. 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 (V_z) that is either positive (e.g. $+2000$ ft.mn^{-1}) or negative (e.g. -2000 ft.mn^{-1}). The vertical speed is zero (*i.e.* 0 ft.mn^{-1}) when the autopilot reaches the target altitude or when the pilot pushes the vertical speed knob to

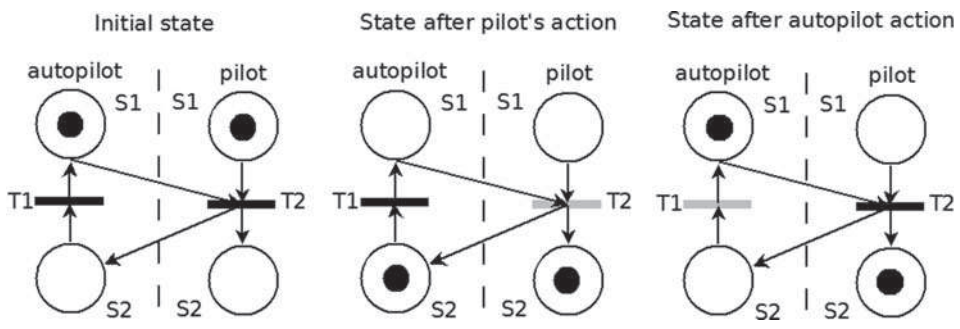

Figure 3. Operator authority limits.

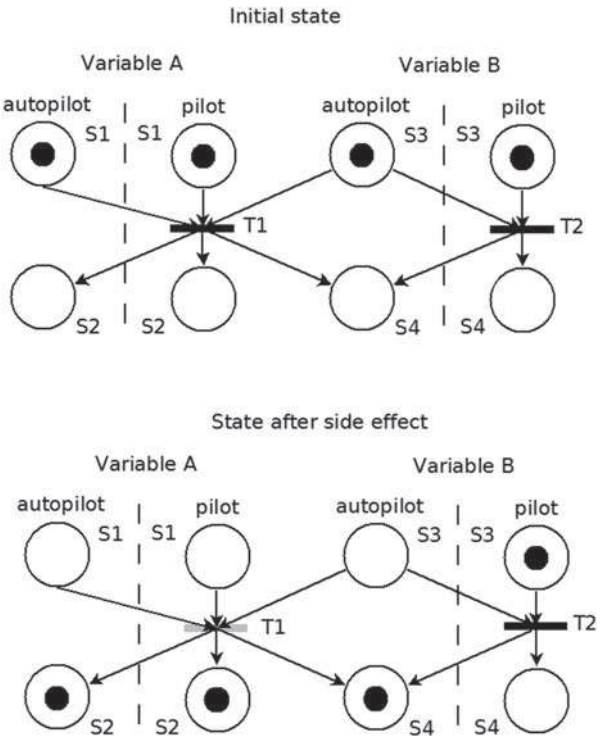

Figure 4. Side effect.

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 (V_{max}) and the vertical speed is zero or negative, then the autopilot climbs at $+1000$ ft.mn^{-1} 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 programming: 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.

3.2 Pattern identification

From the previously described logic, we have modelled the interactions between the pilot and the autopilot with Petri nets (see Figure 5). 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 *automated behaviour* pattern (dashed line boxes). Hereafter, we explain the three of them precisely.

3.2.1 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 (V_{max} – 5 knots).

Initially, the aircraft is levelling off (in Figure 5 *speed: normal*; *vertical speed: level off*; *autopilot connection: ON*; for both autopilot and pilot).

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.

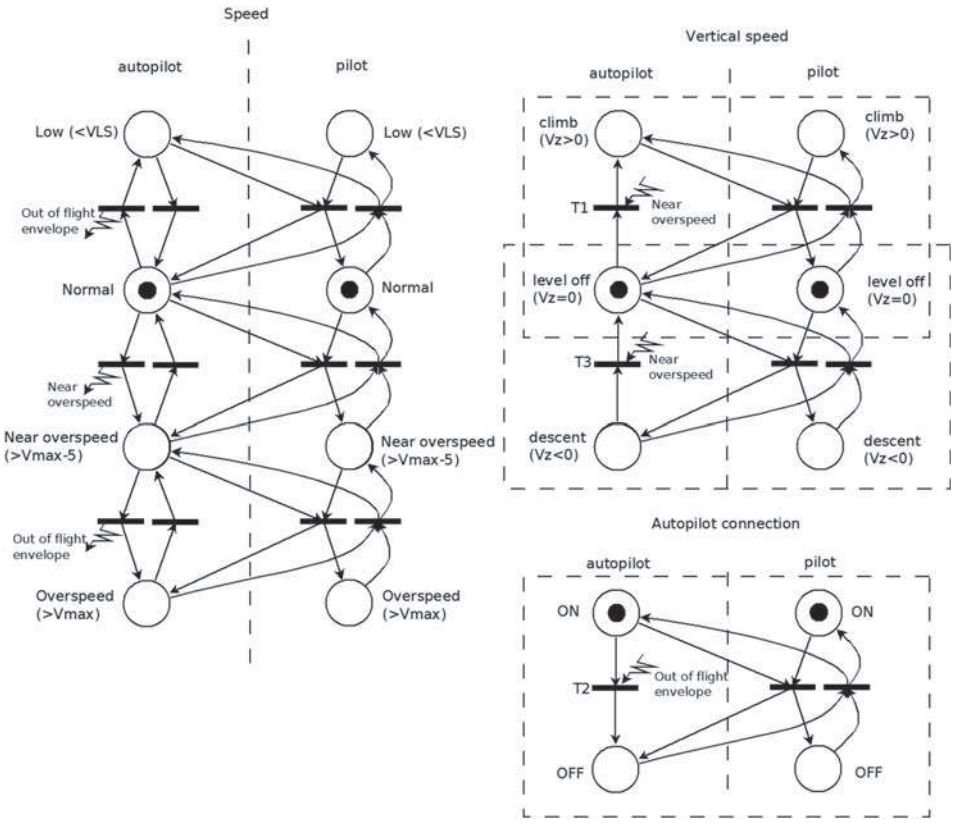

Figure 5. The Petri net model of the autopilot logic.
Note: The boxed parts are the *automated behaviours* pattern instances associated with the dead transitions.

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 FCU. Nevertheless, because of the poor feedback design, we assume that the notification of the mode transition will not be perceived by the pilot.

3.2.2 Autopilot automatic disconnection (T2)

Usually, an aircraft flies near the maximal speed with the autopilot engaged (ON) (in Figure 5 *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 disconnecting the autopilot (transition T2 is fired and the new state is a *utopilot 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 FCU is changed. As two auditory alerts cannot be broadcast at the same time, the autopilot disconnection alarm is inhibited.

3.2.3 Near overspeed mode reversion with inconsistent programming (T3)

Initially, the aircraft is descending (in Figure 5 *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 5, left).

4. Flight simulator experiments

4.1 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 model indeed generate human–machine conflicts in the experimental scenarios, and that these conflicts may remain unnoticed or unsolved by some participants.

4.2 Participants

Ten healthy volunteers (one female; mean age = 38.2 year, SD = 16.3; mean flight experience = 2997.8 h, range = 55–12,000; automated flight desk experience = 314.4 h, range = 10–1000), all French defence staff from Institut Supérieur de l'Aéronautique et de l'Espace (ISAE) campus, were recruited through local advertisement. The participants gave their informed consent after receiving complete information about the nature of the experiment.

4.3 Material: flight simulator

A three-axis motion flight simulator was used to conduct the experiment (see Figure 6). 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 an FCU 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 3.1. As for the autothrust, the logic is as follows:

- The autothrust is engaged and disengaged via a push button 'ATHR' on the FCU. 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.

4.4 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 3.2).

The initial position of the aircraft was on the 14R runway at Blagnac airport (Toulouse, France). The air traffic control (ATC) cleared the aircraft for takeoff and instructed the plane to 'climb 3000 ft, 1500 ft.mn^{-1}'. When the aircraft reached 1000 ft, autopilot engaged, the ATC gave clearance to steer 330° and to increase speed up to 176 knots.

Figure 6. ISAE three-axis flight simulator. Note: Left – outside view of the cabin; right – cockpit view. The arrow indicates the position of the FCU dedicated to program the autopilot (heading, speed, altitude and vertical speed). The participants flew the aircraft from the left seat.

4.4.1 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^{-1} to anticipate possible overspeed (see Section -3.1). This could potentially lead to a collision with the incoming aircraft as the aircraft climbed instead of levelling off as initially required.

4.4.2 Situation 2

When the aircraft reached 1500 ft, it was cleared to 'climb 11,000 ft, 1500 ft.mn^{-1}, steer 322°'. At 10,700 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 11,000 ft could not be achieved anymore and the aircraft kept on climbing with a risk of level bust.

4.4.3 Situation 3

Five minutes later, the ATC instructed the participant to 'slow down, speed 300 knots, descend 5000 ft, 1000 ft.mn^{-1}'. At 9000 ft, the ATC required to 'Accelerate, 325 knots, 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^{-1}. 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 3.1), instead of descending as initially requested.

4.5 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 a 1-h training. They were first trained to perform basic flight manoeuvres 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 programming following the ATC instructions – the participants were instructed to repeat the ATC clearances after programming the FCU, 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 programming, 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 FCU and the different auditory warnings. The 15-mn experimental scenario was then started.

4.6 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 the following – (1) What happened just after takeoff when the ATC required to perform a level-off and what did you do? (2) What happened at 11,000 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 selection, actions on knobs and buttons. The analysis of these data, together with the observations of the experimenters, allowed us to characterise conflict occurrences, conflict detections and conflict solvings. For instance, a conflict situation was characterised by the fact that the participant had failed to manage the vertical separation (the minimum vertical separation between two aircraft is 500 ft).

4.7 *Experimental data processing*

The experimental data were processed so as to characterise the results through four parameters (see Tables 1–3): conflict occurrence; conflict detection; time to first relevant action; and 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 4.1). 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 characterisation of the conflict dynamics, but they are not used for supporting the formal analysis.

5. Results

The results of the experiments are shown in Tables 1–3 for the three situations. A Cochran Q test suggested significant differences regarding the ability of the participants to solve the three different conflicts ($Q(2) = 6.4$, $p < 0.05$).

5.1 *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^{-1}' (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 7).

Only one participant (participant 5, see Table 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.

5.2 *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 500 ft above or under the levelling off altitude (see Figure 8).

All the participants detected the conflict (see Table 2) and seven of them solved it.

5.3 *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^{-1}' to 'Level Off' (T3). In this case, the conflict remained unsolved if the participant flew more than 500 ft above or under the desired descent trajectory (see Figure 9).

Table 1. Near overspeed mode reversion (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 2. Autopilot automatic disconnection (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	Yes	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

Subject 10 did not correctly execute the ATC speed instruction and never flew 'near overspeed' (see Table 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.

6. Discussion

In this study, we propose a generic formal analysis 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 10 general aviation pilots. The results of the experiment tend to show that the transitions that we a-priori identified as critical with the formal model 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, eight out of nine 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 programming 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 (Rome et al. 2012). In the second situation, all the participants detected the conflict and only three 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 18 s for half of the pilots to solve the conflict and to stabilise 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

Table 3. Near overspeed mode reversion with inconsistent programming (T3).

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

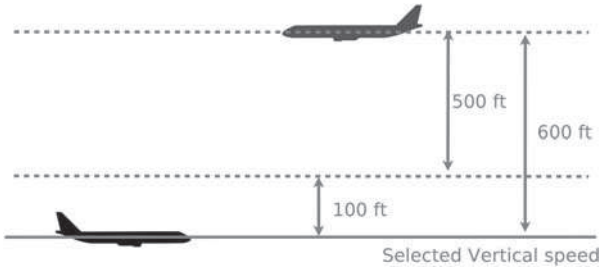

Figure 7. Conflict solved criterion for T1.

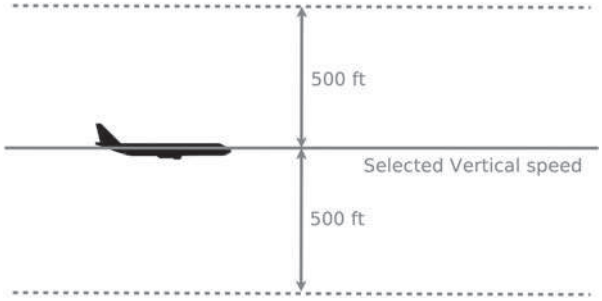

Figure 8. Conflict solved criterion for T2.

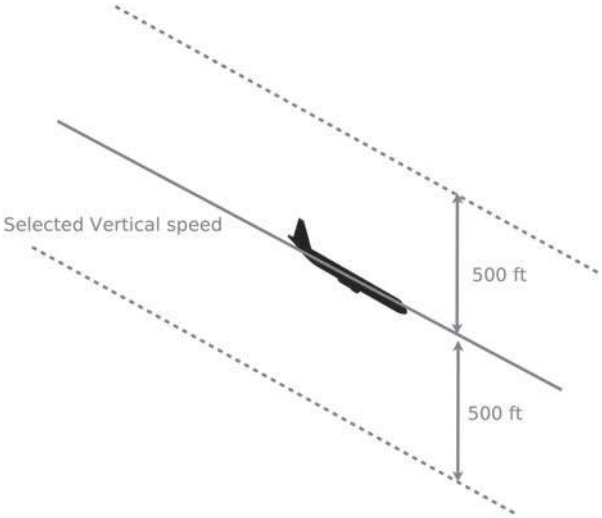

Figure 9. Conflict solved criterion for T3.

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 (Sarter, Mumaw, and Wickens 2007) 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 approach 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 who are not as experienced as transportation pilots to deal with automation.

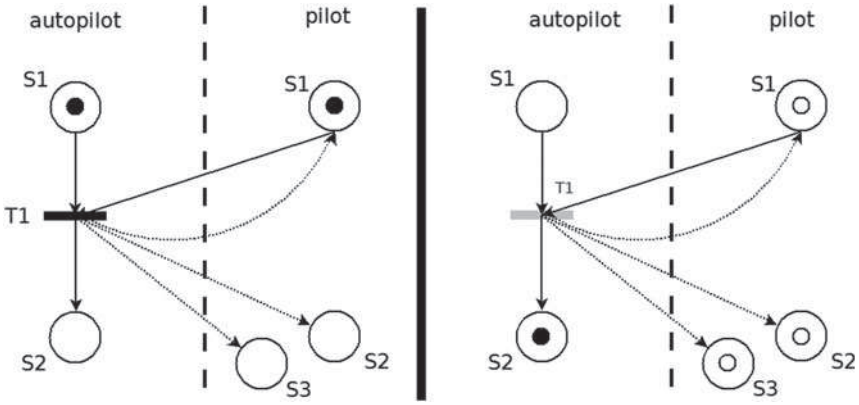

Figure 10. Uncertainty on pilot's situation awareness modelled as a pseudo-firing.

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, then 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 disconnecting 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 user 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 (so there are no false positives). We have no guarantee on the completeness, i.e. whether the formal model detects all the possible conflicts. 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 – for that purpose further work will focus on an approach that can manage uncertainty. 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 (Dubois and Prade 1990) based on fuzzy Petri nets is currently under study to model such ambiguous situations through pseudo-firings (Cardoso, Valette, and Dubois 1999) (see Figure 10): the firing of T1 (autopilot action) leads to the uncertain marking for the pilot's situation awareness: it can be S2 (consistent state) or S1 (pilot unaware of the feedback) or any other state S3 (misinterpretation of the feedback).

The possibilistic model under study is based on three assumptions about the pilot's mental model: (1) it is compliant with the crew operation manual; (2) the pilot could possibly lose feedbacks; and (3) the loss of feedbacks is more likely to happen than slips and errors. A more sophisticated model is to take into account possible misunderstandings of the feedbacks (place S3 in Figure 10) that would be learnt from a pilot's error history database. Thanks to the same kind of database, a mental model that could be non-compliant with the crew operation manual could also be designed. Note that those non-compliances should be potential and not deterministic as for Rushby, Crow, and Palmer (1999), i.e. we should always keep as possible the scenario in which the reception and interpretation of feedbacks are correct. Therefore, each time the pilot makes an error the model should determine *which is the feedback in the past history that, if missed, could explain the execution of this erroneous action as a nominal one*. This model is currently under study and needs further development in order to be tested in real-time missions.

References

Beringer, D. B., and H. C. Harris Jr. 1999. "Automation in General Aviation: Two Studies of Pilot Responses to Autopilot Malfunctions." *The International Journal of Aviation Psychology* 9 (2): 155–174.

Billings, E. 1996. *Aviation Automation: The Search for a Human-Centered Approach*. Mahwah, NJ: Lawrence Erlbaum.

Butler, R. W., S. P. Miller, J. N. Potts, and V. A. Carreno. 1998. “A Formal Methods Approach to the Analysis of Mode Confusion.” Paper presented at the 17th Digital Avionics Systems Conference, Seattle, WA, October 31–November 7.

Cardoso, J., R. Valette, and D. Dubois. 1999. “Possibilistic Petri Nets.” *Systems, Man, and Cybernetics* 29 (5): 573–582.

Creissac Campos, J., and M. D. Harrison. 2008. “Systematic Analysis of Control Panel Interfaces Using Formal Tools.” In *Interactive Systems. Design, Specification, and Verification*, edited by T. C. N. Graham and P. Palanque, vol. 5136, 72–85. Berlin, Heidelberg: Springer-Verlag.

Crow, J., D. Javaux, and J. Rushby. 2000. “Models and Mechanized Methods That Integrate Human Factors Into Automation Design.” Paper presented at the International Conference on Human-Computer Interaction in Aeronautics: HCI-Aero, Toulouse, September 27–29.

David, R., and H. Alla. 2005. *Discrete, Continuous, and Hybrid Petri Nets*. Berlin: Springer.

Dehais, F., M. Causse, and S. Tremblay. 2011. “Mitigation of Conflicts With Automation: Use of Cognitive Countermeasures.” *Human Factors* 53 (5): 448–460.

Dehais, F., C. Tessier, and L. Chaudron. 2003. “GHOST*:* Experimenting Conflicts Countermeasures in the Pilot’s Activity.” Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 164–168, Acapulco, August 9–15.

Dehais, F., C. Tessier, L. Christophe, and F. Reuzeau. 2009. “The Perseveration Syndrome in the Pilot’s Activity: Guidelines and Cognitive Countermeasures.” Paper presented at the 7th International Working Conference on Human Error, Safety, and System Development (HESSD), 68–80, Brussels, September 23–25.

Dubois, D., and H. Prade. 1990. “An Introduction to Possibilistic and Fuzzy Logics.” In *Readings in Uncertain Reasoning*, edited by G. Shafer and J. Pearl, 742–761. San Francisco, CA: Morgan Kaufmann.

Endsley, M. R. 1999. “Level of Automation Effects on Performance, Situation Awareness and Workload in a Dynamic Control Task.” *Ergonomics* 42 (3): 462–492.

Feary, M. 2005. “Formal Identification of Automation Surprise Vulnerabilities in Design.” PhD thesis, Cranfield University.

Hoca, J.-M., M. S. Youngb, and J.-M. Blossevillec. 2009. “Cooperation Between Drivers and Automation: Implications for Safety.” *Theoretical Issues in Ergonomics Science* 10 (2): 135–160.

Holbrook, J., J. Orasanu, and C. McCoy. 2003. “Weather-related Decision Making by Aviators in Alaska.” Proceedings of the 12th International Symposium on Aviation Psychology, 576–581, Dayton, OH, April 14–17.

Inagaki, T. 2003. “Automation and the Cost of Authority.” *International Journal of Industrial Ergonomics* 31 (3): 169–174.

Javaux, D. 2002. “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* 75: 147–165.

Lesire, C., and C. Tessier. 2005. “Particle Petri Nets for Aircraft Procedure Monitoring Under Uncertainty.” Proceedings of the Applications and Theory of Petri Nets Conference, 329–348, Miami, FL, June 20–25.

Leveson, N. G., and E. Palmer. 1997. “Designing Automation to Reduce Operator Errors.” Paper presented at the IEEE International Conference on Systems, Man, and Cybernetics, 2, 1144–1150, Miami, FL, October 12–15.

Orasanu, J., N. Ames, L. Martin, and J. Davison. 2001. “Cognitive and Contextual Factors in Aviation Accidents: Decision Errors.” In *Linking Expertise and Naturalistic Decision Making*, edited by E. Salas, and G. A. Klein, 209–225. Mahwah, NJ: Lawrence Erlbaum.

Parasuraman, R. 2000. “Designing Automation for Human Use: Empirical Studies and Quantitative Models.” *Ergonomics* 43 (7): 931–951.

Pizziol, S., C. Tessier, and F. Dehais. 2012. “What the Heck Is It Doing? Better Understanding Human-Machine Conflicts Through Models.” CEUR Proceedings of the 1st Workshop on Rights and Duties of Autonomous Agents (RDA2) 2012, Vol. 885, 44–49, Montpellier, August 28.

Rome, F., G. Adam, J. Condette, M. Causse, and F. Dehais. 2012. “Go-around Manoeuver: A Simulation Study.” European Association for Aviation Psychology Conference, Sardinia, Italy, September 22–26.

Rushby, J. 2002. “Using Model Checking to Help Discover Mode Confusions and Other Automation Surprise.” *Reliability Engineering and System Safety* 75 (2): 167–177.

Rushby, J., J. Crow, and E. Palmer. 1999. “An Automated Method to Detect Potential Mode Confusions.” Paper presented at the 18th AIAA/IEEE Digital Avionics Systems Conference, St Louis, MO, November 17–19.

Sandys, S. D., S. Koga, N. G. Leveson, L. D. Pinnel, and J. D. Reese. 1997. “Analyzing Software Specifications for Mode Confusion Potential.” Workshop on Human Error and System Development, 132–146, Glasgow University, Glasgow, March 19–22.

Sarter, N. B., R. Mumaw, and C. Wickens. 2007. “Pilots’ Monitoring Strategies and Performance on Automated Flight Decks: An Empirical Study Combining Behavioral and Eye-Tracking Data.” *Human Factors* 49: 347–357.

Sarter, N. B., and D. D. Woods. 1995. “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 (1): 5–19.

Sarter, N. B., D. D. Woods, and C. E. Billings. 1997. “Automation Surprises.” In *Handbook of Human Factors and Ergonomics*, edited by G. Salvendy. 2nd ed, 1926–1943. New York: Wiley.

Weyer, J. 2006. “Modes of Governance of Hybrid Systems. The Mid-Air Collision at Ueberlingen and the Impact of Smart Technology.” *Science, Technology & Innovation Studies* 2 (2): 127–149.

Wiener, E. L., and R. E. Curry. 1980. “Flight-deck Automation: Promises and Problem.” *Ergonomics* 23 (10): 995–1011.

Woods, D. D., and N. B. Sarter. 2000. “Learning From Automation Surprises and ‘Going Sour’ Accidents.” In *Cognitive Engineering in the Aviation Domain*, edited by N. Sarter, and R. Amalberti, 327–353. New York: Lawrence Erlbaum.