What the heck is it doing? Better understanding Human-Machine conflicts through models.

S. Pizziol, C. Tessier, F. Dehais

What the heck is it doing? Better understanding Human-Machine conflicts through models

Sergio Pizziol[1,2] and **Catherine Tessier**[1] and **Frederic Dehais**[2]

Abstract.

This paper deals with human-machine conflicts with a special focus on conflicts caused by an "automation surprise". Considering both the human operator and the machine autopilot or decision functions as agents, we propose Petri net based models of two real cases and we show how modelling each agent's possible actions is likely to highlight conflict states as deadlocks in the Petri net. A general conflict model is then be proposed and paves the way for further on-line human-machine conflict forecast and detection.

1 Introduction

There is a growing interest in unmanned vehicles for civilian or military applications as they prevent the exposure of human operators to hazardous situations. As the human operator is not embedded within the system [22] hazardous events may interfere with the human-machine interactions (e.g. communication breakdowns and latencies). The design of authority sharing is therefore critical [8] because conflicts between the machine and the human operator are likely to compromise the mission [14, 23]. Interestingly these findings are consistent with research in aviation psychology: crew-automation conflicts known as "automation surprises" [18, 19] occur when the autopilot does not behave as expected by the crew (e.g. the autopilot has disconnected and the pilots, who are not flying, are not aware of that [12]). These situations can lead to accidents with an airworthy airplane if, despite the presence of auditory warnings [1], the crew persist in solving a minor conflict [2] "instead of switching to another means or a more direct means to accomplish their flight path management goals" [26].

In this paper we will consider the human-machine system as a two-agent system (see figure 1), i.e. the human agent (the operator) and the automation agent (the autopilot or the embedded decision and planning functions). Indeed both agents can perform actions so as to control the physical system, which may be subject to uncontrolled events (e.g. failures). Notice that an autopilot is considered an agent because some mode changes can be performed by the autopilot itself without prior consent of the pilot, and sometimes despite the pilot'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 operator or the machine is not followed anymore, or the operator has a wrong situation awareness [24], or both. In order to prevent a mission degradation, the agents' plans, and possibly the authority allocation (i.e. which agent is controlling what), have to be adapted [11]. 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.

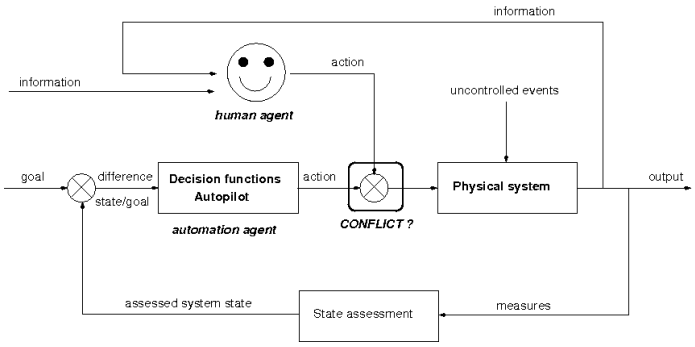

Figure 1: A human-machine system as a two-agent system

We define a conflict as the execution of globally (i.e. at the system level) incoherent actions i.e. one action tends to take the system to state S_a and another one tends to take it to state S_b, and $S_a \neq S_b$. Locally (i.e. at the single agent level) the actions may be coherent with a local plan and the conflict may come from a wrong interaction between the agents. If one agent's local actions are incoherent (e.g. because of a failure) either a local diagnosis and reconfiguration are possible; or they are not (e.g. human operator's error) and the wrong behaviour of this agent is likely to create a conflict with the other agent. Actions in a multi-agent system [9] are incoherent if:

- Physically [21, 20]: at least a depletable or not shareable resource[3] 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. The thrust is a limited resource and may be not enough to grant the climbing rate required by the first agent and the turn rate required by the second one.*
- Epistemically [21]: 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 at 6000 ft and the other one*

[1] ONERA(France) name.surname@onera.fr
[2] ISAE(France) name.surname@isae.fr

[3] As resource we generically refer to a physical object, information, task, goal.

at 4000 ft.

• Logically [20]: at least two goals are logically contradictory, the agents 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.*

Conflicts are situations where incoherent actions, or their consequences, matter in terms of mission achievement, safety, etc. [21, 5]. We distinguish three classes of conflicts that are directly inspired by the classification of incoherent actions: logical conflicts, physical conflicts and knowledge (epistemic) conflicts. Logical conflicts are when the agents' goals are logically contradictory and a trade-off must be found. Note that the goals are not necessarily incompatible: an agent's incapability to accept a trade-off could lead to a conflict. Game theory techniques have been proposed to solve this case of conflict [10]. Physical conflicts are when the agents' goals are independent but incompatible because of the resources required to achieve plans and actions that are associated to the goals, therefore a wise resource sharing is needed. Knowledge conflicts are when the agents' goals are coherent [25, 20], and the agents' information for decision-making about how to reach the goals is not the same. Such conflicts may concern agents' beliefs, knowledge, procedures, opinions.

This paper focuses on knowledge conflicts in human-machine systems, especially the conflicts caused by "automation surprises". Section 2 will focus on two real cases of "automation surprise". Our approach is to assess whether a formal model of those cases could give us avenues for automatic conflict identification and detection. Petri nets (see Appendix) have been chosen for formal modelling since they are well suited to scripted domains with a state dynamics linked to discrete events. From those two cases, we present a generalized conflict model (section 3).

2 What the heck is it doing?

This section presents two real cases of human-machine conflicts caused by "automation surprises", i.e. the machine agent not behaving as expected by the human agent. The first case – a "kill–the–capture" surprise with an MD–88 autopilot has been reported by [13] and investigated by [17, 16]. The second case occurred during an experiment campaign involving one of Onera's Ressac VTOL UAVs[4] in July 2011. For both cases we will show that modelling the agents' possible actions (i.e. what they have the right to do, especially the right to take over the authority from the other agent) enables the conflict to be identified in a formal way. Both cases will be modelled with Petri nets.

2.1 The kill-the-capture surprise

The two agents involved are the Autopilot of the MD-88 and the Pilot. The actions that are considered are the mode transitions of the Autopilot that are triggered either by the Autopilot-agent or by the Pilot-agent. Unlike Rushby [16], we do not make any assumption about a "mental model" of the Pilot, but we take the objective viewpoint of what the Pilot actually does. For the sake of clarity only the relevant modes and mode transitions are represented. In our Petri nets, we use the same colour code as in [17]: green for done by the Pilot, red for done by the Autopilot

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

In the Initial state *Alt-Capture* mode of the Autopilot is not armed (initial marking "Alt-Capture not Armed") – figure 2.

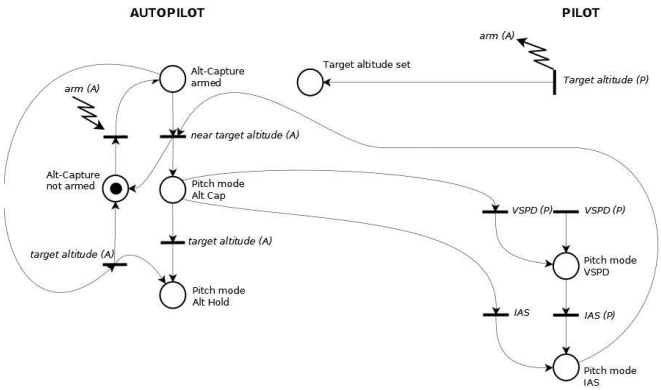

Figure 2: Alt-Capture not Armed

The Pilot sets altitude to *Target altitude*. 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), then to *IAS* (Indicated Air Speed – climb rate adjusted, constant air speed) – figure 3.

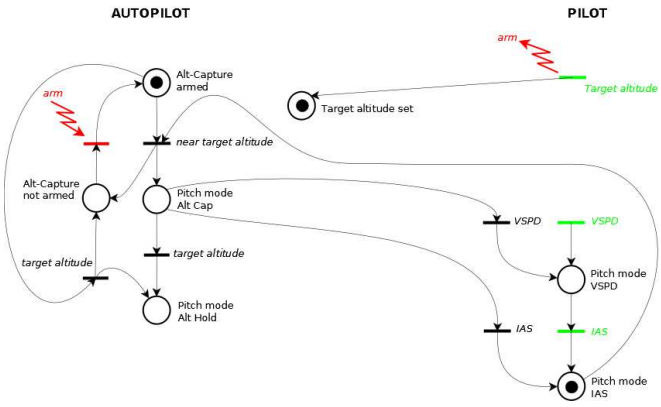

Figure 3: Alt-Capture armed and IAS

When target altitude is nearly reached, the Autopilot changes Pitch mode to *Alt Cap* (provides smooth levelling off at the desired altitude) therefore mode *Alt-Capture* is disarmed, so as *Pitch mode IAS* – figure 4.

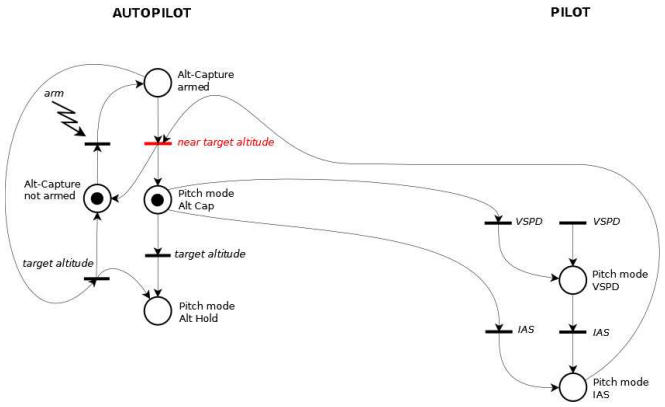

Figure 4: Alt Cap; Alt-Capture disarmed

The Pilot then changes Pitch mode to *VSPD*, therefore *Pitch mode Alt Cap* is disarmed – figure 5.

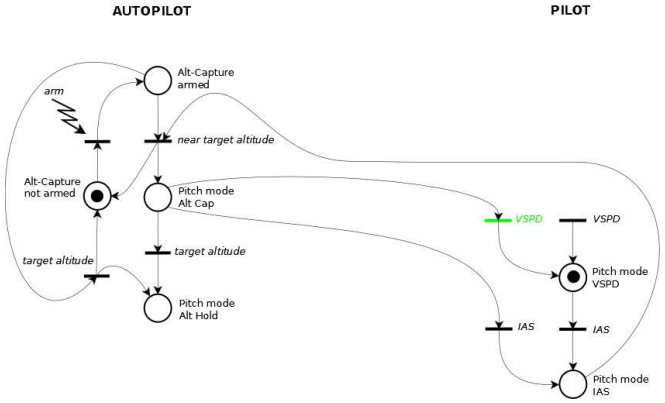

Figure 5: Pitch mode VSPD

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 6.

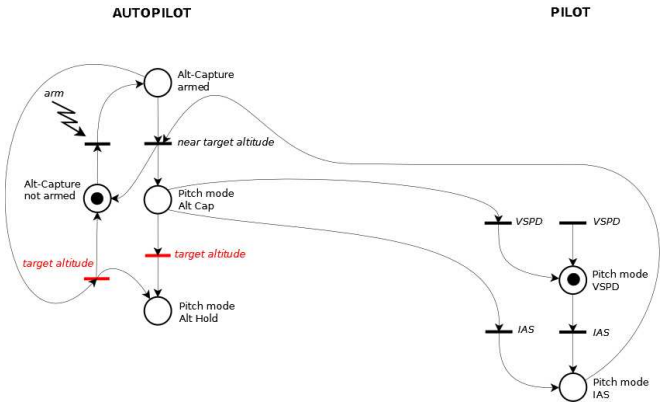

Figure 6: Event target altitude lost – "Oops, it didn't arm" [13].

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 on the Autopilot side (i.e. no transition can be fired anymore). This is a *knowledge conflict* [21] as the consequences of the agents' actions were neither assessed properly nor explained to one another.

2.2 Rain and automation

The second case of "automation surprise" 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 "automation surprise". 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 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). This last mode can be engaged only by the Safety pilot who has always pre-emption rights, activating an exclusion switch cutting off the automatism. 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 achievement
Nominal autopiloting- Autonav	*				*
Nominal autopiloting- Operator flight plan	*	*	*		*
Nominal Manual- high level	*		*		
Nominal Manual- assisted	*		*		
Emergency Manual				*	

Table 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 actions (transitions) and states (places) of the Ressac software agent (right) and of the human operator agent, i.e. what happens on the Gp's interface and the possible actions of the Sp (left). The procedure to follow (see figure 7 left) matches the plan (see figure 7 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 procedure is stopped. Initial state is human agent and software agent both in the 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 software agent. This transition emits Event B (information waypoint alpha reached displayed on the Gp interface) which updates the procedure: human agent state is Phase 2, so as software agent state.

Phase 2/ Phase 3 operates the same way with Event C (waypoint beta) and D (information displayed on the Gp interface and procedure updated).

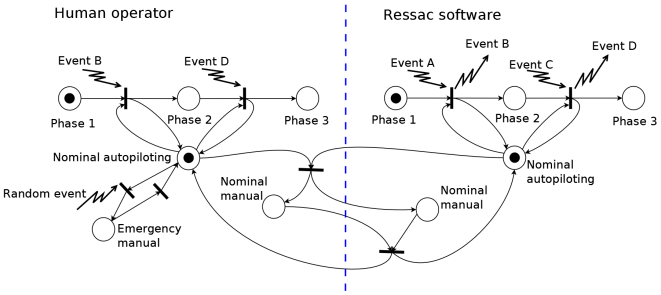

Figure 7: Initial state

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 8 transition Random event is fired by the human agent and Emergency manual place is marked.

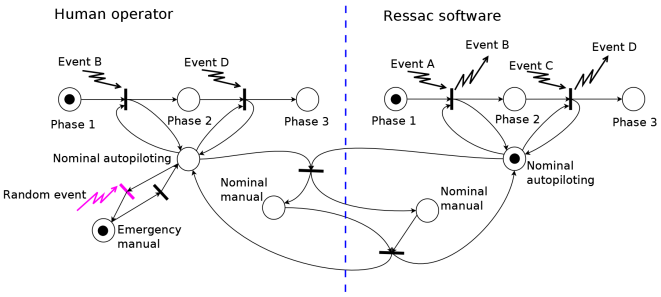

Figure 8: Rain and emergency manual mode

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 9).

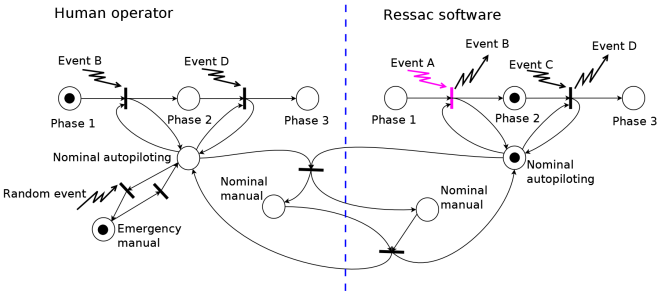

Figure 9: Software state update

Event B is emitted but lost on the human agent side, since one precondition (Nominal autopiloting) is no longer verified (figure 10).

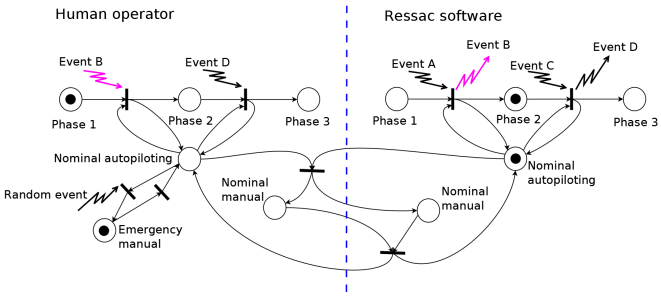

Figure 10: Lost of the event for the procedure update

The rain stopped and the Sp decided that the nominal plan could be resumed. Transition Emergency manual to Nominal autopiloting is fired (figure 11). The nominal plan was resumed (Phase 2) and Ressac headed waypoint beta. The human operators, who were expecting Phase 1 to be resumed, did not understand what Ressac was doing and began to panic. This is again a *knowledge conflict* [21] in which the human operators considered the behaviour of the machine as a failure. Indeed none of the test team members properly interpreted the behaviour of Ressac.

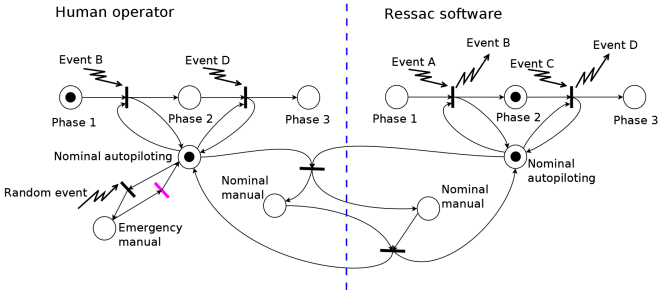

Figure 11: What the heck is it doing?

Notice that the marking of the Petri net (figure 11) is such that: (i) place Phase 2 is marked on the software agent side whereas place Phase 1 is marked on the human agent side ; (ii) one place Nominal piloting is marked (software agent side) whereas the other one is not marked (human agent side). Nevertheless it is a matter of semantic inconsistencies and not of formal inconsistencies within the Petri net model. Indeed for case (ii), the two places Nominal piloting do not represent the same state, otherwise a unique place would have been used: one is the software agent state and the other one is the human agent state.

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, detection and solving.

3 Towards a model of human-automation conflict

3.1 Conflict model

In a multi-agent system different agents are often interested in the knowledge of the same state variables. Those variables can seman-

tically describe the physical environment state or the agent internal state. The values of those state variables can be affected by the agents' actions.

Let us consider two agents A1 and A2 that both have the right to act on a common device to change its state. The state of the device must be successively S1 then S2 and the agents must always have the same knowledge about the device state. The initial state is S1 In figure 12, both agents' knowledge is the same, i.e. the device state is S1 (left). The result of the firing of T1 is that both agents' knowledge is that system state is S2 (right). Note that transition T1 represent a synchronization of both agents about their shared decision.

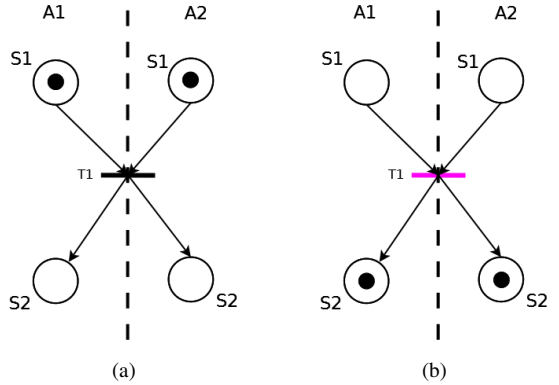

Figure 12: Two-agent system, correct design

As far as figure 13 is concerned, A2 need A1 to fire transition T2, i.e. both agents' knowledge must be S1 to make the device evolve to S2. On the contrary the firing of transition T1 only makes A1's knowledge state evolve to S2 (transition T1 is "hidden" from A2)(left). If T1 is fired, the result is that A1's knowledge is S2 whereas A2's is S1 and transition T2 is dead (right). This is a conflict.

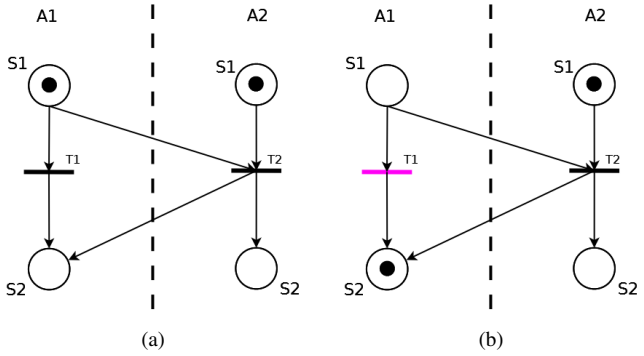

Figure 13: Two-agent system, incorrect design

3.2 Conflict solving

In figure 13 T1 is a 'hidden transition" so far as agent A2 cannot see it neither the consequences of its firing. That is the case for the "Rain and automation" example, figure 10.

Two solutions are then possible. The first one is to remove T1, i.e. agent A1 has no right to fire T1. In this case we get the ideal case in figure 12, we allow only *shared decisions* represented by transition T1. The second solution is to *inform* A2 of the firing of T1, see figure 14.

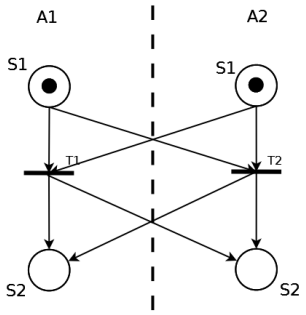

Figure 14: Two-agent system, another correct design

If A2 is a human operator the effect of a transition on his knowledge is not sure: the feedback he receives from the other agent can be lost or misinterpreted. A pseudo-firing [3] for T1 can model this kind of uncertainty, see figure 15 (left). The firing of T1 leads to the uncertain marking for the agent A2 state represented in figure 15 (right) by empty markers.

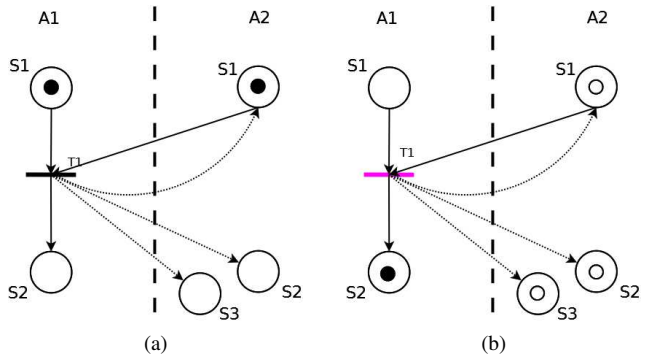

Figure 15: Two-agent system, A2 is human. Pseudo firing on correct design

For that reason the second solution proposed (inform the other agent) has an uncertain effect if A2 is human. This kind of transition is considered as a vulnerability by some researchers [7]. In other works the not nominal effect of a transition can be restored informing the human operator again or differently [6].

4 Conclusion and further work

Starting from two real cases of "automation surprises", we have shown that a formal model allows us to characterize a Human-Machine conflict: for both cases the Petri net model features a deadlock (i.e. at least one transition cannot be fired). We have then proposed a general Petri net based conflict model that paves the way for automatic conflict detection through "hidden" transitions identification and liveliness properties checking. We have also given two possible design solutions to prevent conflicts: *share the decision* or *inform the other agent*.

Nevertheless if the agent being informed is human the problem of the correct reception and interpretation of the information has to be considered. Therefore uncertainty has to be modelled so as to feed an estimator of the human agent's knowledge state: such an estimator,

which is further work, can be based on the human agent's actions and "internal state" [15].

Current work focuses on further aircraft autopilot-pilot interaction modelling – especially some cases that led to accidents – so as to put to the test the generic conflict model we have proposed. The next steps will be on-line conflict forecast and detection and experiments in our flight simulator.

5 Appendix: Petri Nets

A Petri net $< P, T, F, B >$ is a bipartite graph with two types of nodes: P is a finite set of places; T is a finite set of transitions [4]. 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. Petri nets allow sequencing, parallelism and synchronization to be easily represented.

References

[1] 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**(2), 155–174, (1999).

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

[3] J. Cardoso, R. Valette, and D. Dubois, 'Possibilistic Petri nets', *Systems, Man, and Cybernetics*, **29**(5), 573 – 582, (1999).

[4] R. David and H. Alla, 'Discrete, continuous, and hybrid petri nets.', (2005).

[5] F. Dehais and P. Pasquier, 'Approche générique du conflit.', *ERGO-IHM*, (2000).

[6] 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, year = 2009, volume = , number = , pages = .*

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

[8] T. Inagaki, 'Automation and the cost of authority', *International Journal of Industrial Ergonomics*, **31**(3), 169–174, (2003).

[9] Nicholas R. Jennings, 'Controlling cooperative problem solving in industrial multi-agent systems using joint intentions', *Artif. Intell.*, **75**(2), 195–240, (1995).

[10] S. Kraus, 'Negotiation and cooperation in multi-agent environments', *Artif. Intell.*, **94**(1-2), 79–97, (1997).

[11] S. Mercier, C. Tessier, and F. Dehais, 'Authority management in human-robot systems', *11th IFAC/IFIP/IFORS/IE Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, (2010).

[12] R. Mumaw, N. Sarter, and C. Wickens, 'Analysis of pilots' monitoring and performance on an automated flight deck', in *Proceedings of the 11th International Symposium for Aviation Psychology*, Columbus, OH, USA, (2001).

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

[14] R. Parasuraman and C. Wickens, 'Humans : Still vital after all these years of automation', *Human factors*, **50**(3), 511–520, (2008).

[15] S. Pizziol, Fr. Dehais, and C. Tessier, 'Towards human operator "state" assessment', in *ATACCS'2011 - 1st International Conference on Application and Theory of Automation in Command and Control Systems*, Barcelona, Spain, (2011).

[16] J. Rushby, 'Using model checking to help discover mode confusions and other automation surprise', *Reliability Engineering and System Safety*, **75**(2), 167–177, (2002).

[17] J. Rushby, J. Crow, and E. Palmer, 'An automated method to detect potential mode confusions', in *18th AIAA/IEEE Digital Avionics Systems Conference*, St Louis, MO, (1999). Presentation slides.

[18] 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**(1), 5–19, (1995).

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

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

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

[22] A.P. Tvaryanas, 'Visual scan patterns during simulated control of an Uninhabited Aerial Vehicle (UAV)', *Aviation, space, and environmental medicine*, **75**(6), 531–538, (2004).

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

[24] C.D. Wickens, 'Situation awareness: review of Mica Endsley's 1995 articles on situation awareness theory and measurement', *Human factors*, **50**(3), 397–403, (2008).

[25] R. Wilensky, 'Planning and understanding: A computational approach to human reasoning', in *Reading, MA: Addison-Wesley.*, (1983).

[26] D. Woods and N. Sarter, 'Learning from automation surprises and going sour accidents', in *Cognitive engineering in the aviation domain*, eds., N. Sarter and R Amalberti, 327–353, Lawrence Erlbaum, New York, (2000).