2014 IEEE International Conference on Systems, Man, and Cybernetics
October 5-8, 2014, San Diego, CA, USA

Checking formal verification models for human-automation interaction

M. M. (René) van Paassen
Aerospace Engineering
Delft University of Technology
Delft, The Netherlands
M.M.vanPaassen@TUDelft.nl

Matthew L. Bolton
Department of Industrial and Systems Engineering
State University of New York at Buffalo
Buffalo, NY, USA
m.l.bolton@buffalo.edu

Noelia Jiménez
IXION Industry and Aerospace
Madrid, Spain
NJimenez@IXION.es

***Abstract*—In complex human-machine systems, unforeseen failures in the interaction between human agents, automation and the environment provide an important contribution to incidents and accidents. The complexity of many systems precludes a designer from foreseeing all possible states of interaction. Formal verification methods are explored as a means of making human-machine systems more robust against failures arising from these unforeseen interactions. For these methods, a analytic model of the operator task is combined with a formal model of the system (automation), and, using model checking tools, a formal verification of the interaction is performed. Validity of the results, however, does require a sufficient correspondence between the model and the actual system. To validate this correspondence, this study explores an approach where the predictions from a formal model are compared to behavior of the human-machine system. The Paparazzi UAV ground control station is used as a test case, and a framework was created to automatically play back results from the formal verification tool to the UAV ground control simulation. The results show a good correspondence between the actual system and the model results, even if the model is by necessity a simplified description of actual system behavior. A remaining problem is creating enough variation in verification tool traces to properly test the correspondence between the formal model and the system.**

***Foreword* – This work is based on the outcomes of the project "Verification Models for Advanced Human-Automation Interaction in Safety Critical Flight Operations" sponsored by the European Space Agency**

***Index Terms*—Ergonomics, human factors, interfaces, automation, formal verification**

I. Introduction

Human error is cited as a major contribution in failures in aerospace systems. A part of these failures arise as unforeseen and unwanted interactions between a human operator and automation. The complexity of many systems prevents designers and analysts from foreseeing and manually verifying all possible states in human-automation interaction. While different tools and methods are aimed at creating robust interaction in design [1], [2], [3], analysts can still miss problems because most analysis approaches are incapable of evaluating all of the possible ways in which humans and automation interact. However, formal verification techniques, and particularly model checking, offer means of performing such robust analyses.

A. Formal verification and model checking

Formal methods are mathematically robust languages, techniques, and tools for the modeling, specification, and verification of systems. A formal model is a mathematical description of the behavior of a system. Using software tools, such a model may be checked for desirable or undesirable "properties", providing a mechanized way of proving these properties. The properties can generally be specified using temporal logic, which combines Boolean logic on the model variables with temporal logic operators. In LTL, properties may, for example, be used to describe properties that must globally hold and thus always be true (G).

B. Formal verification of HAI

The application of formal verification is still largely limited to software and electronic hardware. However, for human-automation interaction (HAI), the same model verification tools can be used. The challenge here is however to create a proper model that can be validated by automated model checkers. This work uses an approach developed by Bolton et al. [4], [5], [6], in which the human operator model is specified in Extended Operator Function Model (EOFM). The model for the automation and controlled device must be written directly in the model checking tool's language, in this case SAL [7]. A more complete overview of the use of formal methods in HAI verification is given in [8].

C. Verification of model validity

Formal verification is commonly constrained by the limitations of formal verification tools. Complex models for formal verification can contain a large number of states, and generating proof for a property requires the tool to explore this potentially vast state-space. Especially continuous variables are hard to capture in formal models. Although a version of SAL provides the capability of checking models with continuous states, currently the EOFM language and code generator are not able to exploit this. Verification models for systems that include continuous state variables are thus of necessity a – somewhat coarse – approximation. In addition, there is the problem of verifying that the formal model for a system is actually a correct model; errors in the formal verification model

978-1-4799-3840-7/14/$31.00 ©2014 IEEE

could lead to false positive results or missed results. In the project "Verification Models for Advanced Human-Automation Interaction in Safety Critical Flight Operations", which was aimed at evaluating and extending the application of formal verification to HAI, three sets of test cases were used to verify the methodology. The third test case was based on the open-source Paparazzi UAV ground control station [**?**] and it was used to explore model verification accuracy.

The remainder of this paper describes the Paparazzi test case, the formal verification approach taken for this test case and the means by which the correspondence between the formal model and the actual system are checked.

II. Formal verification approach

The method for formal verification used here is based on the approach developed by Bolton and Bass[5], [6]. For this approach, human operator behavior is modeled using the Extended Operator Function Model (EOFM, itself an extension from OFM[4]). The EOFM model is a hierarchical description of the operator's activities. Each activity again may consist of several sub-activities, or of one or more actions. Actions are atomic operator task elements, and commonly represent the influence on the controlled system, modeling e.g. button presses or other inputs. Activities and actions may be given activation conditions, completion conditions, and in case an activity or action can be repeated, repetition conditions. The software described in [5] can take an EOFM model, expressed in XML notation, and generate code representing this model in the SAL[7] model checking language.

The EOFM model must then be combined with a model for the controlled system. This model is commonly specified in SAL directly, as a set of state transitions that model reaction to operator actions or a natural transition ("dynamics") in the system itself.

III. Paparazzi test case

The Paparazzi system is a generic and flexible control system for unmanned aircraft. Both Vertical Take-Off and Landing (VTOL) aircraft (such as quad-rotors and helicopters) as well as horizontal take-off aircraft can be controlled. The system uses a ground control station that runs on a Linux PC (typically Ubuntu, since for that operating system all software is packaged and readily available). The ground control station can connect to one or more UAV's, and it can also connect to simulations of the UAV's, mainly for training and for verification of the feasibility of flight plans. When connected to a simulation, the system is also a representative micro-world task for supervisory control operation of a dynamic system.

A. Modeled elements

The UAV test case for formal verification describes the interaction of an operator with a single UAV through a control station implemented in Paparazzi[9]. The operator's task is to perform a surveillance flight with the UAV.

Two different surveillance task elements in the flight are defined. One of these task elements is a perimeter surveillance that, in the actual UAV set-up, automatically completes. For example, after performing the surveillance, the UAV returns to a standby waypoint. The other task is an area surveillance task. When not interrupted, this latter task automatically repeats and the operator must actively abort this task when it has been completed.

The battery capacity of the UAV should allow for a flight with both elements present if the initial battery level is high enough and the weather, specifically wind conditions, permit this. The challenges of this application for formal verification with EOFM and SAL are the following:

- The Paparazzi application represents a dynamic process. To match this process with a verification model, many of the dynamics – specifically the travel of the UAV in 3 dimensions, and the battery depletion – must be represented in the model. In this case, battery level and flight progress are represented in a limited number of discrete values.
- To further check the ability of the verification models to represent the actual dynamics, the traces generated by the model checker are interpreted by a program that in turn controls the Paparazzi simulation to replay these traces. Besides providing a means to verify the extent to which the dynamics have been accurately represented, this provides an illustrative way of showing the different traces generated by the model checker.

B. Operator model

The operator of the UAV needs to perform several tasks for the operation. At start-up, the proper connection between the UAV and the ground station must be verified and one must wait for proper navigation and sensor signals. Weather conditions, and principally wind conditions, must be verified. Further, the launch direction and initial climb out point might need to be adjusted if too much crosswind or tailwind is present. For launch, the UAV engine must be switched on and the launch command given.

After a successful climb to the initial climb out point and when the UAV systems perform properly, the route for perimeter monitoring or the route for survey monitoring may be started. At all times performance and battery capacity must be monitored to ensure proper functioning and safe return of the vehicle. After the monitoring flight, further area surveillance is possible through the placing of the surveillance waypoints and the selection of area surveillance.

Return and landing is performed by verification of the approach fix and touchdown waypoints and selection of either a left hand or right hand downwind approach. The transition to a new flight path element takes place automatically, if that is so specified in the flight path. However, this is normally done on command from the operator. The transition is largely automated and manoeuvres that are needed to, e.g. position the UAV for starting a surveillance flight, are automatic.

The operator model is based on a task breakdown for the operator's mission, two main tasks are identified:

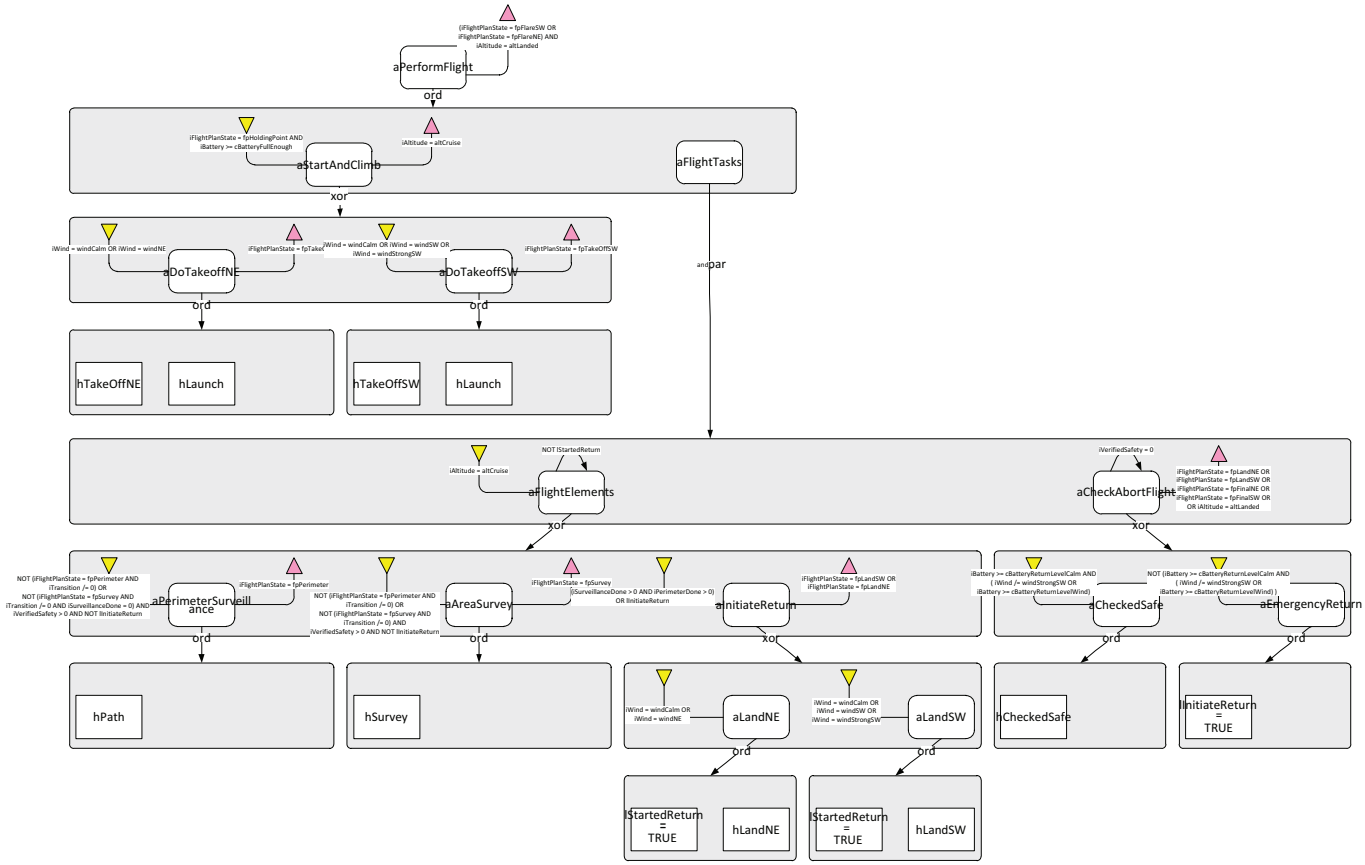

Fig. 1: Visualization of the final EOFM model for the UAV operator tasks.

aPerformFlight Perform a surveillance flight. This task is completed when the UAV has again landed at the proper location. Several sub-tasks can be distinguished, these task are performed in sequence:

aStartAndClimb This describes the start and initial climb out phase of the flight. This task can be initiated when the interface indicates that the UAV is ready (its initial checks are complete), and the operator should verify that the battery level is sufficient. The take off may, depending on the wind, be in a North-Easterly (60 degrees) or South-Westerly (240 degrees) direction. To model the possibility of choosing the proper take-off direction, the task is decomposed into two sub-tasks, of which only one is executed.

aDoTakeoffNE Initiates a take-off on 06

aDoTakeoffSW Initiates a take off on 24

The sub-tasks are completed once the flight management system is in the proper mode, leaving the operator model to handle subsequent tasks.

aFlightTasks This describes the flight tasks that need to be performed. This task can be initiated once the previous task **aStartAndClimb** has completed. The tasks in flight consist of the surveillance element, and of a continuous monitoring of battery level for assessing flight safety. Those two tasks must both be performed and should be performed in parallel.

aFlightElements These are the "normal" flight elements. They may be performed in any order, and the activity is repeatable, enabling the model to repeatedly run an element. The last element is the initiation of the return to the base, this element stops the repetition of the **aFlightElements** activity.

aPerimeterSurveillance This is the flight of a path along the terrain's perimeter. In the example scenario it is defined by four waypoints. The conditions for starting the track are a sufficient battery level and not being busy in performing another task element. When the perimeter has been checked, the UAV automatically returns to the StandBy waypoint. The task is complete – freeing up the operator model to initiate other tasks – once the UAV model has started the surveillance.

aAreaSurvey This element is a zig-zag pattern flight over an area defined by two waypoints. The same conditions as for the previous task apply. When the survey is complete, the UAV would automatically repeat the survey, without action from the operator.

aInitiateReturn This is the return procedure, to

be initiated when both the above tasks are completed, or when the battery level is not sufficient any longer to perform one of these tasks.

aCheckAbortFlight As already mentioned, parallel to the normal flight tasks, is a check on the battery level that will initiate an abort when needed. This check is repeated while the aircraft has not landed. It triggers one of two sub-activities:

aCheckedSafe A verification that the battery level is sufficient for continuation.

aEmergencyReturn Set a flag internally in the operator model that will enable the aInitiateReturn activity, while disabling the other flight elements in **aFlightElements**.

The model is summarized in Figure 1.

C. UAV Model

The UAV and interface behavior model implemented in SAL updates a number of state parameters that describe the state of the UAV and are the basis for the interaction with the operator model.

iFlightPlanState This is an enumerated value that describes the state of the flight management system.

iEngine A Boolean value that indicates whether the engine is switched on or off.

iBattery The fill level of the battery, expressed in 0.1 V increments. Ranges from 89 (battery dead, engine off) to 125.

iAltitude The altitude, discretized into altGround or altLanded and altLow, altCruise.

iPerimeterDone Indicates the number of times the perimeter has been flown.

iAreaDone Indicates the number of times the surveillance area has been covered.

iTransition Indicates the number of remaining steps (linked to battery decrements, and representing time), that the UAV needs to fly in order to complete the current element in the **iFlightPlanState**.

iVerifiedSafety Indicates the remaining validity (also linked to battery decrements and representing time) for the last check of the battery level.

With the exception of the last, all these states represent an observable property in the Paparazzi UAV operator interface (see Figure 2). An overview of the flight plan logic is given in Figure 3. A particular challenge encountered during development concerned parallelism in execution. The operator model contained a main activity that described how the different flight tasks were to be initiated, and a second activity that described the monitoring of the battery level. These two activities were specified in the operator model as parallel activities. However, that by itself does not guarantee parallel execution in the SAL/EOFM model. To force the two activities to interleave, a virtual "dead-man's" button was implemented in the UAV model. The battery level monitoring activity was modified to

Fig. 2: Links between the EOFM/SAL model and the Paparazzi ground station interface. Model variables are linked to their approximate location for perception of the UAV and ground station output (variables starting with "i") or their action button (variables starting with "h").

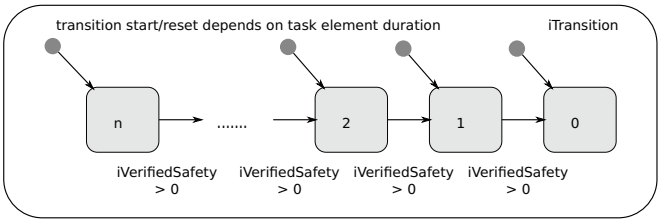

Fig. 4: Transitions describing battery progress in the flight and the dead-man's button trick.

activate this "dead-man's" button in its monitoring cycle, and the autonomous transitions of the UAV model would depend on the regular activation of the "dead-man's" button, which resets **iVerifiedSafety** to its initial value (Figure 4). With each step in flight progress and battery decrement, the **iVerifiedSafety** parameter is also decremented. When this parameter reaches zero, the UAV model transitions are blocked and the operator model is forced to re-evaluate the battery level before other actions can be taken and before the UAV model progresses. In a manner, this resembles a dead-man's button, which is not actually implemented in the UAV system, and is needed here to force a transition for the operator model to monitoring the battery level. The value of the **iVerifiedSafety** parameter indicates what the remaining time span is, in battery decrement cycles, for which the operator can avoid checking the battery level.

The model for the UAV and its interface, as implemented in SAL, contains three sets of transitions:

- Transitions that describe the reaction to a user intervention. These in general switch to a different flight plan state, e.g. to simulate the response to a user pressing a different flight element button (actions "hTakeOffNE", "hTakeOffSW", "hLaunch", "hPath", "hSurvey",

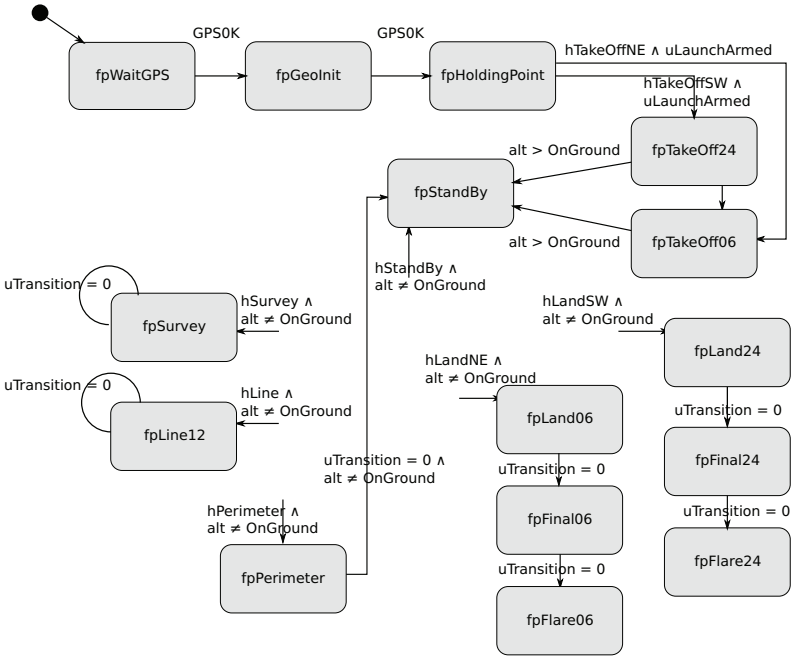

Fig. 3: Transitions between the different flight plan states.

"hLandNE" and "hLandSW")

- Autonomous transitions of the model, e.g. after completion of the perimeter, the UAV will autonomously proceed to the standby waypoint, while after completion of the survey, the survey is repeated.
- Updates of the time. This is performed by a single transition, that updates all of the **iVerifiedSafety**, **iTransition** and **iBatteryLevel** parameters. The first parameter indicates whether the last observation by the operator is still valid, while the second one indicates the progress of the UAV in a flight segment. The meaning of the **iBatteryLevel** parameter may be obvious from its name.

IV. Model validation results

The intent of the UAV application is to test the HAI verification method with a model that (1) represents a dynamic system, and (2) to test the possibility for automatically comparing the model traces predicted by the verification tool and the actual system behavior.

A. Formal model verification results

The base UAV model assumes that the check interval is set at 5 time increments, corresponding to approximately 90 seconds. In the analysis, the initial battery level was varied between empty (8.9V) and full (12.5V), but the operator model would only initiate a flight for a battery level of 12V or higher. The operator model initiates a return when the battery level drops below 10.2V.

Using the SAL model checker, combined human operator and UAV model's behavior was verified. Several properties were tested:

- A property that asserted that the UAV would always land at the proper landing site. A counterexample would mean that the UAV lands not at its landing spot. For the above parameters, this property was proven by the model checker. After an increase of the battery check interval or of the duration of the approach and landing flight segments, this property produces a counterexample, representing an off-site landing, indicating that the verification procedure can check this property.
- Properties to determine the possibility to repeat flight elements, demonstrating the flexibility of the model. It proved to be possible to perform the area surveillance 3 times, and the flight along the perimeter 2 times. These properties were asserted negatively, and thus the counterexample would indicate the flight was possible.

Using the formal model, it is possible to "tune" the operator model, effectively creating a procedure, that return flight is initiated on time. Variation of the conditions in the EOFM model, with either a longer interval for the battery check, a different battery level for the return decision violate that condition.

B. Correspondence between verification model and simulation

In a next step, the verification model is compared to the Paparazzi simulation application. The model is checked by attempting to replicate the HAI interaction traces predicted by the formal model – in the form of counterexamples – and using these to drive the Paparazzi UAV station with a simulated UAV. To get a proper coverage, a variety of traces were generated and tested **(NORM)** ; a normal flight **(NE,SW,SW+)** ; flights with varying wind strength, which affects the flight times for different segments **(AREA)** ; a flight in which the order of task elements was changed from the normal flight, with area surveillance first

TABLE I: Comparison between the EOFM/SAL formal model traces for several flight conditions, and the corresponding Paparazzi simulation.

Trace	S-BAT	P-BAT	remarks
NORM:	10.2	10.6	Duration of flight elements corresponded well. There was no wind (this was not forced in the property, so any type of wind could have resulted.).
AREA:	10.6	10.2	Wind was calm.
NE:	10.3	10.1	Wind as specified.
SW:	10.6	10.2	The paths are shorter for this flight than for the landing towards the north-east
SW+:	10.2	10.0	The SAL/EOFM model is essentially the same, but transition times for strong wind are used. The wind does some influence on the duration of the flight.

A python script was written to parse the SAL counterexample traces. The traces contain information on all variables in the model, including the state of the human operator actions and the UAV model state. For the evaluation the Paparazzi interface and a UAV simulation are started. The actions were converted into button clicks on the UAV control station interface. The relevant model changes were compared to the progress of the UAV; the replay script would wait until these changes were observed in the Paparazzi simulation before progressing with subsequent user inputs in the counterexample.

All counterexample traces generated by the model checker could be re-played by the replay program with the Paparazzi UAV simulation and Ground Control Station. The counterexamples show the decrements of the battery level, in steps of 0.1 V. By trial and error, it was determined that a single decrement occurs after approximately 18 seconds of engine use. For the construction of the SAL model, the duration of different flight segments was determined in a hand-flown simulation, and these were then incorporated in the SAL model for the UAV.

A summary of the final battery levels (which in the simulation and SAL model are in 0.1 V increments, but shown in V here) are given in Table I.

C. Computation times

Time required for verification of the model is reasonable. On an Intel i7 with 8 GB ram, generating a proof or counterexample for a single property typically takes in the order of a minute. Checking all properties in the UAV model (automatically generated and manual, in total 191 properties) takes approximately two hours.

V. Results and Conclusions

In the development of the operator and system model for the UAV application, a specific modeling approach was used. To mimic the dynamic behavior of the UAV, a model transition is generated for discrete increments, corresponding to the rate of battery voltage depletion. One decrement in battery voltage (corresponding to 0.1 Volt) is approximately equivalent to 18 seconds. This implies that transitions, the mechanism to represent state change in SAL, take place asymmetrically between the operator model (which has fewer transitions) and the UAV and interface model (which performs most).

Replaying a SAL counterexample trace with the Paparazzi UAV simulation and Ground Control Station proved to be feasible. All generated traces were correctly played back, with relatively small differences in battery level predicted by the SAL trace and the one resulting from the simulation. The replay program has elements specific for controlling the Paparazzi simulation – since a Paparazzi specific protocol was read from the IVY bus, but the principle can be re-used for other application that can be controlled using mouse clicks and and of which the output can be observed.

The modeled operator behavior can also be used as a template for procedures to be used in the UAV operation. The formal model checking supports the verification of these procedures and of the criteria used in making decisions, such as the battery level at which the return to the landing spot should be initiated.

It is recognized that the check on correspondence between the formal model and the Paparazzi UAV application is not a substitute for a formal verification. However, the use of a variety of different traces and the matching results do provide a level of confidence in the model.

References

[1] K. J. Vicente, *Cognitive Work Analysis. Toward a Safe, Productive, and Healthy Computer-Based Work*. Mahwah, NJ: Lawrence Erlbaum Publishers, 1999.

[2] T. B. Sheridan and R. Parasuraman, "Human-automation interaction," *Reviews of human factors and ergonomics*, vol. 1, no. 1, pp. 89–129, 2005.

[3] N. Leveson, *Engineering a safer world: systems thinking applied to safety*, ser. Engineering systems. Cambridge, Mass: MIT Press, 2011.

[4] M. L. Bolton and E. J. Bass, "Enhanced operator function model: A generic human task behavior modeling language," in *Proceedings of the IEEE International Conference on Systems Man and Cybernetics*. Piscataway: IEEE, 2009, pp. 2983–2990.

[5] M. L. Bolton, R. I. Siminiceanu, and E. J. Bass, "A systematic approach to model checking human-automation interaction using task-analytic models," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 41, no. 5, pp. 961–976, 2011.

[6] M. L. Bolton and E. J. Bass, "Using model checking to explore checklist-guided pilot behavior," *International Journal of Aviation Psychology*, vol. 22, no. 4, pp. 343–366, 2012.

[7] L. De Moura, S. Owre, and N. Shankar, "The SAL language manual," Computer Science Laboratory, SRI International, Menlo Park, Tech. Rep. CSL-01-01, 2003.

[8] 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: Systems*, vol. 43, no. 3, pp. 488–503, 2013.

[9] "Paparazzi - The free autopilot," http://paparazzi.enac.fr/wiki/Main_Page, 2013, accessed November 2013.