

Example of a Complementary use of Model Checking and Agent-based Simulation

Gabriel E. Gelman
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, USA
g.gelman@gatech.edu

Karen M. Feigh
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, USA
karen.feigh@gatech.edu

John Rushby
Computer Science Laboratory
SRI International
Menlo Park, USA
rushby@csl.sri.com

Abstract—To identify problems that may arise between pilots and automation, methods are needed that can uncover potential problems with automation early in the design process. Such potential problems include automation surprises, which describe events when pilots are surprised by the actions of the automation. In this work, agent-based, hybrid time simulation and model checking are combined and their respective advantages leveraged in an original manner to find problematic human-automation interaction (HAI) early in the design process. The Tarom 381 incident involving the former Airbus automatic speed protection logic, leading to an automation surprise, was used as a common case study for both methodology validation and further analysis. Results of this case study show why model checking alone has difficulty analyzing such systems and how the incorporation of simulation can be used in a complementary fashion. The results indicate that the method is suitable to examine problematic HAI, such as automation surprises, allowing automation designers to improve their design.

Index Terms—simulation, model checking, automation surprise, mental model, formal methods

I. INTRODUCTION

One failure of human-automation interaction (HAI) that exhibits significant risk to aviation is when automation behavior deviates from what pilots expect. This effect is called “automation surprise” (AS) and has been examined by Wiener [1], Sarter and colleagues [2] and Palmer [3]. A common example of AS in aviation is when the aircraft flight mode automatically changes and the pilot is neither aware of the change, nor able to comprehend the behavior resulting from the flight mode, i.e. “mode confusion”. Javaux [4] concluded that mode confusion appears because pilots are not aware of all the preconditions for a given mode change. Javaux’s work explained how pilots simplify or reduce the number of preconditions that trigger a certain mode change to only the preconditions that are salient. Preconditions for intended mode changes that are not salient, but are almost always satisfied, are forgotten. Circumstances in which the forgotten precondition is not satisfied often lead to mode confusion.

The challenge presented by AS is that the automation is operating *as designed*. Yet, the flight crew is still unable to predict or explain the behavior of the aircraft. It is clear that the problem lies with a mismatch between the pilot’s expectation,

which is driven by his/her mental model of the autoflight system, and the underlying control logic. Norman [5] has done fundamental work in the human-computer interaction domain, such as HAI, of mental models. To be able to investigate system behavior and predict AS and other potential problems that may arise between collaborators in such systems, a method for designers is needed that investigates not just autoflight system validation, but the pilot’s understanding on how the particular design works.

Two methods for analyzing system behavior and detecting potential problems in HAI are: model checking (MC) as done by Rushby and colleagues [6]–[8] and agent-based simulation as done by Corker and colleagues [9]. MC and simulation are two distinct analysis methods, both having their advantages and limitations. MC originated in computer science as a technique to check if a finite-state system (the model) satisfies a given specification, usually written in a temporal logic. If the specification is not satisfied, the model checker can construct a counterexample that drives the model through a scenario to manifest the violation.

MC technology has advanced considerably and is no longer limited to finite-state systems; so-called “infinite bounded model checkers” can handle models that include mathematical integer and real values, and checkers for hybrid systems allow these values to be related by differential equations. The appeal of MC as an analysis technique is that it considers *all possible* behaviors of the model, so it is *certain* to find a violation of the specification if one exists, and can *prove* correctness otherwise.

A limitation of MC is that the underlying technology is automated deduction, where all problems are at least NP-hard, and so the model cannot be too large or too complicated if the analysis is to complete in a reasonable time. Consequently, the full or “concrete” model must often be considerably simplified before it can be model checked. If the simplified model is a true “abstraction” of the concrete model (i.e., includes all its behaviors) then safety properties proved of the abstraction are also true of the concrete model; the converse may not be so, however, for a property that is true of the concrete system may fail in MC because the abstraction is too coarse (i.e., the model has been oversimplified, leading to spurious counterexamples).

To apply MC to HAI problems, we compose models of the automation (often a complex state machine), the controlled plant (typically a hybrid system whose dynamics are described by differential equations), and human expectations about future states, and look for divergences between expectation and true state. To make the analysis feasible, the models of the plant and of the automation usually must be drastically simplified (e.g., time may not be accurately modeled, or the plant may even be omitted altogether), which can result in a spurious counterexample. The counterexample must then be examined to see if it suggests a concrete scenario that manifests a genuine problem, or is spurious. If it is spurious, the abstraction must be refined to eliminate the responsible oversimplification, and the whole process repeats.

Due to the difficulties of constructing effective abstractions, interpreting counterexamples, and refining abstractions to eliminate spurious counterexamples, analysis by MC has been impractical for all but the most basic HAI problems [10].

Thus, HAI has been more generally analyzed using a combination of human-in-the-loop studies and agent-based simulation where the human agents are modeled to capture the aspects of human perception and cognition of interest to the study at hand. Prime examples of such an agent-based modeling approach used in the aviation domain are the MIDAS simulation created by Corker and colleagues [9] and WMC (Work Models that Compute) as described in [11], [12]. Agent-based simulations have the ability to use a variety of timing mechanisms (continuous, discrete event, or hybrid) to account for the dynamics of the systems under analysis, and generally include hundreds of states for when all agents are accounted. Agent-based simulations, however are limited to exploring one trace through the state-space per “run”. Thus to more fully explore the system multiple simulation runs are required. Common simulation analysis techniques such as Monte Carlo methods attach probability distributions to key input variables and then randomly sample from the distributions to perform thousands of simulation runs. Simulations can never exhaustively explore the state-space that their models represent, and thus cannot be used to prove the safety of the modeled system. Simulations can provide more information about the events leading to problems in HAI and work with sophisticated and realistic models of human agents, automated agents and their environment. However, simulations depend strongly on the assumptions made, as well as the quality of the modeling and input data.

MC and agent-based simulation have complementary strengths and weaknesses: MC can examine all possible scenarios, but only for simplified models; simulation can examine realistic models, but only a fraction of the state-space. A method that leverages the strengths of both could allow a much more complete and systematic evaluation of realistic models than current practice. The objective of this work is to evaluate a common scenario using both methods, compare and contrast the outputs and inputs, and examine the assumptions and requirements for the modeling. More generally, such approaches can be used to study future autonomy and authority

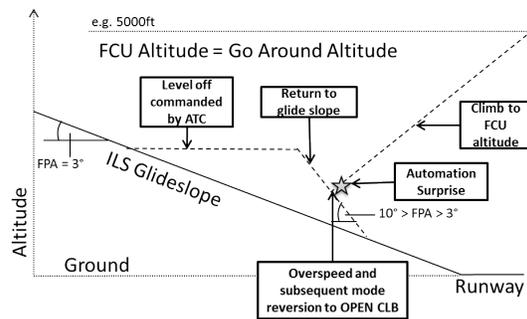


Fig. 1. Exemplary scenario in which AS occurs.

assignments between pilots and automation, the risks such new assignments bear and ultimately how to mitigate these risks. Finally, the ability to use two distinctive methods jointly could have implications on methods for examining human-integrated systems in general.

II. EXAMPLE SCENARIO: A320 SPEED PROTECTION

This work uses a known AS found in previous versions of Airbus models A300, A310 and A320 as a case study. An unexpected automatic flight mode change during approach was initiated by the automatic speed protection logic. The result is a mode change from vertical speed / flight path angle (V/S FPA) mode, in which the aircraft maintains either vertical speed or flight path angle at a certain value, to an open mode which gives priority to the airspeed rather than the vertical speed or flight path angle. Two open modes are available, open climb (OPEN CLB) or open descend (OPEN DES). The open modes ignore any flight path angle or vertical speed previously commanded and rely on the altitude entered in the Flight Control Unit (FCU). If the FCU altitude is set to an altitude above the current altitude, the automation switches into the OPEN CLB flight mode and starts climbing. In descent this mode change results in an AS to the pilots who are attempting to land. The most prominent example of where this flight mode logic almost lead to an accident was Tarom flight 381 involving an Airbus A310, traveling from Bucharest to Paris Orly in 1994. The mode reversion to OPEN CLB caused the aircraft to climb steeply on approach, stall and lose over 3,000ft in altitude before the pilots recovered at 800ft.

To examine the benefits and limitations of the methodology presented in this work, the Tarom flight 381 scenario was modified to serve as an example scenario. The sequence of the modified scenario examined in this work is shown in Figure 1. In this scenario, the pilots are told by Air Traffic Control (ATC) to level off while on the ILS glideslope (G/S) and subsequently are forced to descend with a higher FPA to recapture the G/S when ATC clears them to resume course. ATC may issue such an instruction if the airspace is congested or the runway is still occupied. Such a scenario is easy to implement and it is realistic. The longer the aircraft is commanded to hold the altitude, the steeper the FPA is required to regain the G/S, and the more the aircraft accelerates.

This incident is interesting as it involves the correct functioning of autoflight control logic, yet resulted in unacceptable behavior of the joint human-automation system. The control logic acted in such a way as to surprise the pilots, and in the case of Tarom Flight 381, for the pilots to fight the autoflight system until the aircraft stalled. It is this interaction of autoflight logic, context, and human variability that form the difficulties in identifying and analyzing HAI problems.

A key challenge in modeling AS, such as the AS linked to the described Airbus speed protection logic, is the inclusion of the pilot's mental model. Here, the mental model comprises two constructs: expectations and beliefs. The pilot's expectations about the behavior of the aircraft are based on the pilot's belief about the aircraft's current flight mode and his expectation of the aircraft behavior resulting from this mode. The pilot's expectations are based on his observations about the current state of the aircraft, previous actions and mental model, i.e. understanding of how the aircraft will react to commands given its current state. When the pilot knowingly commands changes, then those changes are expected and anticipated.

The scenario described above was modeled in MC using the SAL (Symbolic Analysis Laboratory) suite of model checkers from SRI [13]. The implementation is thoroughly described in [14], and will only be described briefly here. The model contained three components: `automation` (the autopilot and autothrottle), `pilots` (all aspects of human-automation interaction) and `airplane` (the dynamics of the aircraft with its engines) and seven state variables: `flap_config`, `vertical_mode`, `mental_model` (enumerated types), `speedvals`, `altvals`, `thrustvals`, and `pitchvals` (reals with bounded ranges), and two constants: `VMAX` `speedvals` maximum and `Vfe` `speedvals` maximum with flaps extended. Additionally, a `constraints` module is specified to model the aircraft dynamics and is modeled as a synchronous observer to restrict the model checking to those states which do not violate the constraints, i.e. only inspect states which represent realistic aircraft dynamics. A final module in the form of another synchronous observer is added to seek out the AS, should it occur. The AS is flagged by this module when the expectation (here: mental model) does not correspond to an aspect of the state of the aircraft anymore. In this case, when the pitch deviates from the pilot's expectation about the pitch.

The example scenario was also modeled in the hybrid-time agent-based simulation framework, WMC. Specifically, WMC is used to model the aircraft dynamics, autoflight and flight management system logic, the flight crew and ATC. Similar to MC, a mental model for human agents is required in WMC to simulate an AS. In contrast to SAL, WMC has more flexibility with actions that are performed by the agents. The structure of WMC makes it simpler to create the mental model aspect of beliefs about the current state of the aircraft. The beliefs can be updated in the background every time the human agent performs an action where he perceives state changes in his environment. However, simply taking a copy of the state of the world every timestep and updating the belief system is not

sufficient to detect an AS. These beliefs are updated regularly and are therefore always in sync with reality and provide no way to detect an AS. The human agent's expectations aspect of the mental model (in SAL simply called the mental model) must also be modeled. Such an expectation construct needs to formulate the agent's expectation about the aircraft's future state based on the actions he performs. Four basic aircraft state variables are used to compute pilot expectation: altitude, airspeed, vertical speed and heading. AS result from the pilot's expectations being violated about these variables.

To formulate expectations, the aircraft variables that cause the changes to the basic aircraft variables were defined. Different flight modes follow different guidance logic. Depending on the flight mode there are different target values the aircraft tries to maintain or achieve, which includes airspeed, vertical speed, altitude, latitude, longitude, throttle, heading, flight path angle and timing values. The guidance logic a flight mode follows determined the expectation formulation.

III. METHOD

The objective of this research is to analyze a common scenario using both MC and simulation to illustrate how the two analysis methods could be used in combination. Figure 2 shows one possible method for combining the two analysis techniques. Here, the narrative based on Tarom flight 381 is used as input for a model specification in SAL, which then is exhaustively explored. Since MC can explore all possible states in a simple and highly abstracted model specification, it is used first to find scenarios in which system behavior differs from the expectations of the pilot, such as in the case of AS.

In the next step as shown in Figure 2, the preconditions that the SAL counterexample trace reveals are the necessary preconditions for the examined scenario to occur (such as an AS), are given to WMC. The preconditions describe the states that eventually lead to the examined scenario. The requirements that are necessary to recreate the SAL preconditions are then modeled in WMC. Scenario creation in WMC requires consideration of a significantly larger number of variables (because the SAL models are heavily abstracted), and comparable scenarios must be carefully designed. Specifically, the concrete variables that are involved in making the SAL scenario emerge must be identified and guided suitably in WMC. In most cases, simulations are created that are both normative and deterministic so that they may be checked for errors. Once properly verified, then stochastic elements or off-nominal occurrences or behaviors can be interjected. The simulation then must be run enough times to capture the variability of interest often using either a full factorial design of experiments or a Monte Carlo approach, both of which are likely to require hundreds or thousands of runs. Depending on the complexity of the simulation and speed of the computational hardware this can take several minutes to several days.

In the subsequent step, the results of the scenario runs in WMC provide a more concise description of the preconditions that lead the examined scenario to appear, if the scenario

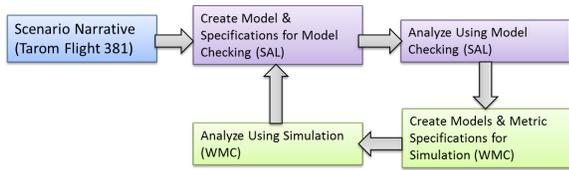


Fig. 2. The two methods, SAL and WMC, and how they are connected in the method.

appears at all. These results will provide feedback on whether the scenario in SAL is realistic and under which conditions it is likely to occur.

Building on SAL’s scenario, WMC provides a realistic scenario if it appears under the preconditions provided by SAL. This scenario includes a more precise description of the preconditions under which the scenario occurs and an outline of the sequence and times of events in the scenario. In the step after, the next iteration of the cycle (Figure 2) is performed. At this point there is a detailed scenario description that includes times of events and states of the aircraft at these times. These preconditions and this sequence is used to make the SAL model more complex, through including more state transitions. Also, adding more variables and more agents (or modules as they are called in SAL), may be necessary.

A realistic scenario has the following properties:

- Preconditions leading to the observed behavior must have the potential to appear in reality.
- Aircraft simulation behavior has to be grounded in reality, for instance mode reversions given in the simulation need to be based on real preconditions.

After completing one round of analysis (Figure 2), several outcomes are possible. First, the problems identified by the model checker have been verified by the simulation and can be referred to a higher fidelity simulation facility, such as a cockpit flight simulator, for further validation. If system behavior that potentially leads to dangerous situations (such as AS) is uncovered, then a solution should be sought to prevent the problem from occurring. Second, it is also possible that the analysis will suggest scenario variants based on the examined system behavior. Third, at the end of any given iteration it may be discovered that a scenario leads to a dead end, meaning that such a scenario is an artifact of the abstraction and does not appear in reality. These spurious scenarios are not worth exploring further and the abstraction used in MC should be refined to eliminate them.

In our case study, a counterexample trace produced by MC was confirmed in simulation, and new potentially dangerous scenarios linked to the automatic speed protection were also uncovered (i.e., the first and second of the possible outcomes described in the previous paragraph). This “local” exploration of variant scenarios was more easily performed in simulation than in MC. To complete the loop (we did not do this), the abstraction used in MC could be modified to represent proposed solutions to the problems discovered and the analysis repeated to verify their effectiveness or reveal further problems.

TABLE I
VARIABLES USED TO DEFINE THE SIMULATION SCENARIOS.

Variable	Description	Values
Go Around Altitude	What is the go around altitude that is set in the MCP	(3000-6000)ft in 500ft increments
Flaps Extension Speed	What is the speed of the aircraft when pilots extends flap	(201-226)kt in 5kt increments
Level Off Altitude	At what altitude is the level off commanded by ATC	(4000-5000)ft in 1000ft increments
Level Off Duration	For how many seconds does the aircraft level off	(30-100)s in 10s increments

IV. SIMULATION ANALYSIS EXPERIMENTAL DESIGN

In contrast to MC where the abstracted model can be exhaustively searched within seconds (the equivalent of running thousands of simulation runs), simulation is restricted to evaluating one path through the state-space included in its model at a time. Thus, care must be taken to determine which subset of the state-space is of interest and to create a design of experiments to explore that space.

The goal of the WMC simulation in this analysis is to:

- 1) Verify that the action sequence predicted by SAL to be problematic continues to be problematic.
- 2) Refine SAL’s prediction to include specific temporal relationships between events.

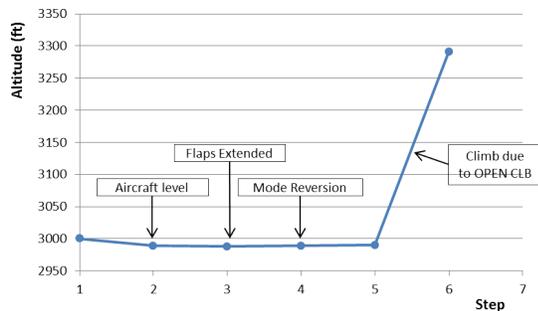
Scenarios were created following a full factorial design of experiments in this case study. All of the variables included in Table I affect the occurrence of an AS linked to the Airbus automatic speed protection logic. To provoke an AS, adequate yet realistic values have been selected. The preconditions that were shown in [14] to cause AS were used for the scenarios; making it possible to see whether these same preconditions also lead to an AS in WMC.

V. RESULTS

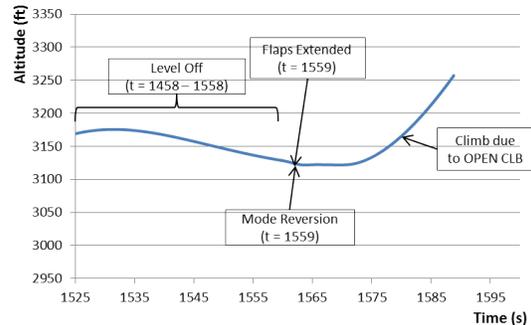
The results of the simulations of the case study verify the results of the Bass et al. [14] analysis holds, and uncover several other potentially dangerous scenarios linked to the automatic speed protection. Specifically, if the automatic speed protection was still implemented in its original form, many different types of potentially dangerous scenarios involving an AS could occur. These are not limited to the way the AS appeared in the Tarom 381 incident.

A. Bass et al. matching case [14]

The WMC dynamic aircraft model (Boeing 747 model) is modified from its standard large transport aircraft to make the Bass et al. (Airbus A310 model) scenario appear with the same or a similar altitude profile. Specifically, the aircraft model was modified to match the maximum speed of the Airbus model used in SAL and the target speeds were adjusted to fit the assumptions made in SAL. The level off was commanded at about 3000ft and flaps extension was forced to be performed after the level off. When these preconditions are given the scenario appears as described in Bass et al. Figures 3(a) and



(a) SAL (from Bass et al. [14])



(b) WMC output

Fig. 3. Altitude profile in SAL and WMC with Bass et al. preconditions. Since SAL does not model time, the x-axis in the left graph is showing the steps, representing time implicitly. The distance between steps is arbitrary. In the right graph the aircraft levels off slightly above 3000ft and after resuming descent, flaps are extended which leads the aircraft to overspeed and change modes to OPEN CLB and climb to the FCU altitude of 3291ft causing an AS.

3(b) compare these two altitude profiles. Since SAL only models discrete states, altitude points are connected by lines.

B. Overspeed Causes

WMC's standard transport aircraft dynamics and operational procedures modeling, waypoints and maximum speeds were used, for the exploration of the state space with the four independent variables. In the Tarom example, there are two overspeed causes. The first is extending flaps at an unsafe speed. The second is diving to capture or recapture the glideslope. In the Tarom incident both causes were present; the aircraft did not capture the glideslope properly requiring a dive to recapture it. During the dive, the pilots extended the flaps prematurely, reducing the aircraft's maximum safe airspeed below that of the aircraft's actual speed.

For the state-space exploration, a level off requested by ATC causing the aircraft to deviate from the glideslope after which the aircraft dives to recapture the glideslope was used to simulate the aircraft not capturing the glideslope as in the Tarom example. In the state-space exploration simulations, overspeeds generated by both the premature extension of flaps and the dive appeared individually, as well as an overspeed as a result of both causes at the same time. The combination of both causes means the aircraft dove to recapture the glideslope, extending the flaps decreased the maximum speed and the dive made the aircraft reach this new maximum speed.

C. Grouping Results into Meaningful Outcomes

In contrast to MC which sought to prove or disprove a specific mode transition could occur and under what preconditions, simulations are rather open ended. A purpose of the WMC simulation was to search for a more generic state of AS arising out of the combination of the independent variables manipulated. The simulations showed three different outcomes, AS involving the OPEN CLB or OPEN DES mode and outcomes without a mode change. These outcomes were grouped based on their causes. Given the number of independent variables and the complexity of an actual simulation comprising flight dynamics, six such outcomes appeared in

TABLE II
OVERVIEW OF THE OUTCOMES THAT APPEARED. (*) MEANS AN AS OCCURRED, AND IT IS UNCLEAR WHETHER THIS IS DUE TO THE SIMULATION GUIDANCE LOGIC OR WHETHER THIS IS REAL BEHAVIOR AND (**) MEANS THAT EVEN THOUGH AN AS OCCURRED, THIS IS DUE TO THE SIMULATION GUIDANCE LOGIC AND UNLIKELY TO APPEAR IN REALITY.

OC	Mode	AS	Description
1	DES	No	Mode Reversion before level off, early
2	CLB	Yes	flaps extension leads to overspeed
3	DES	Yes*	Mode Reversion after level off, early flaps
4	CLB	Yes	extension leads to overspeed
5	DES	Yes**	Mode Reversion after level off, dive leads
6	CLB	Yes	to overspeed on current flaps configuration

the results as described in Table II. Each outcome has its counterpart for OPEN CLB and OPEN DES, representing three stages or reasons for which a mode reversion occurs for each of the modes. The first mode change (Outcomes 1 and 2) appears due to early flaps extension before the level off which sets the actual airspeed instantaneously to be above maximum. The second mode change (Outcomes 3 and 4) occurs after the level off during the dive, for the same reason. The third mode change (Outcomes 5 and 6) occurs during the dive, the pilot extends the flaps at the correct time, however the dive makes the aircraft overspeed.

VI. ANALYSIS

A. Bass et al. Equivalent Scenario

With the SAL inputs at 3291ft go around altitude and flaps extension speed 21kt above maximum speed (226kt in WMC), as given in the Bass et al. scenario, a full spectrum of AS occurred as shown in Figure 4. Here the pattern of outcomes resulting from the SAL preconditions are shown. What is striking is the complex pattern of the interaction that level off duration and level off altitude have on the appearance of different outcomes. This pattern illustrates the difficulty that model checking has in analyzing these types of systems. Of particular interest is Outcome 1 which may be almost invisible to the flight crew as the mode change would be from one

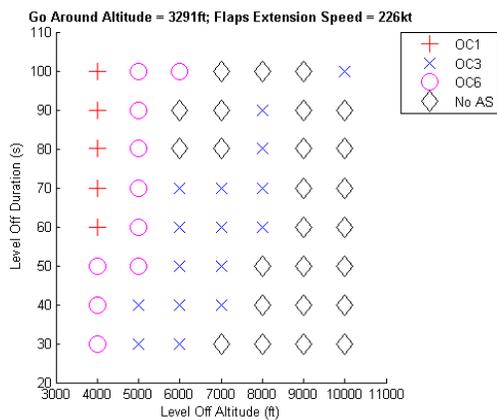


Fig. 4. Outcomes that appeared under the preconditions given in Bass et al.: Go Around altitude of 3291ft and flaps extension speed 21kt above the maximum speed (226kt).

descent mode to another. However depending on the altitude listed in the FCU, the flight path might diverge from that intended and stray outside the safe approach path.

B. Next Iteration: Modelcheck WMC Scenarios

Using the model and scenario given by Bass et al. [14] produces six outcomes in WMC that involve the automatic speed protection (Table II). Of these six, four outcomes elicit an actual AS that could occur in reality. As explained in the methods section in III and as is shown in Figure 2, the next step after obtaining the WMC results is to use the outcomes of interest to make a more elaborate SAL model and to initiate a new iteration. Conceptually, the results from simulation can be used in two ways to improve or further exploit MC, depending on whether or not simulation confirms the HAI problems identified by MC.

- Had simulation shown that the MC produced scenarios were spurious, we could have used the insight from simulation to refine the abstraction used in MC. Using a Counter Example-Guided Abstraction Refinement (CEGAR) approach, we could iterate this process until either a real problem is identified or MC is able to prove that none exist.
- If, as here, simulation confirms the dangerous scenarios discovered by MC (and discovers others), we could either 1) Instruct the model checker to look for counterexamples conceptually different than those found (e.g., to find outcomes different from 14, we could instruct the model checker search for scenarios where the flap extension is fixed), or 2) Modify the abstraction used in MC to incorporate proposed automation design fixes or procedure changes, and verify whether these eliminate the problems.

VII. CONCLUSION

This work provides an example of a complementary method for combining simulation and MC in conjunction to look at potentially problematic aspects of system behavior, such as newly introduced automation functionality or changes in

the authority and autonomy assignment of tasks between controllers and pilots. The method combines the strengths of MC, i.e. exhaustive analysis of large state space, with the strengths of hybrid-time, agent-based simulation, i.e. explicit modeling of system dynamics and time. The method allows a larger set of scenarios to be analyzed than would be possible using conventional means of human-in-the-loop simulation with high-fidelity simulation.

Presently, the method requires a significant amount of manual modeling to translate the output of one analysis method into useful models and input values for the other. Future work will investigate the possibility of creating methods to automate the model creation and modification.

ACKNOWLEDGMENT

This work was funded by NASA award NNA10DE79C and NNA13AB02C and solely the responsibility of the authors and does not necessarily represent the official views of NASA.

REFERENCES

- [1] E. Wiener, "Human factors of advanced technology (glass cockpit) transport aircraft," 1989.
- [2] N. Sarter, D. Woods, and C. Billings, "Automation surprises," *Handbook of human factors and ergonomics*, Vol. 2, pp. 1926–1943, 1997.
- [3] E. Palmer, "Oops, It Didn't Arm. A case study of two automation surprises," in *Proceedings of the Eighth International Symposium on Aviation Psychology*, Columbus, OH, April 1995, pp. 227–232.
- [4] D. Javaux, "Explaining Sarter and Woods classical results. the cognitive complexity of pilot-autopilot interaction on the Boeing 737-EIS," in *Proceedings of Human Error, Safety, and System Development (HESSD98)*, Seattle, Washington, April 1-2 1998.
- [5] D. Norman, "Some observations on mental models," *Mental models*, Vol. 7, 1983.
- [6] J. Crow, D. Javaux, and J. Rushby, "Models and mechanized methods that integrate human factors into automation design," in *Proceedings of HCI-Aero 2000: International Conference on Human-Computer Interaction in Aeronautics*, 2000, p. 163168.
- [7] J. Rushby, "Analyzing cockpit interfaces using formal methods," in *Proceedings of FM-Elsewhere*, ser. Electronic Notes in Theoretical Computer Science, H. Bowman, Ed., Vol. 43. Pisa, Italy: Elsevier, Oct. 2000.
- [8] —, "Using model checking to help discover mode confusions and other automation surprises," *Reliability Engineering and System Safety*, Vol. 75, No. 2, pp. 167–177, Feb. 2002.
- [9] K. M. Corker and B. R. Smith, "An architecture and model for cognitive engineering simulation analysis: Application to advanced aciation automation," in *AIAA Computing in Aerospace 9 Conference*, 1993.
- [10] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Using formal verification to evaluate human-automation interaction, a review," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 43, No. 3, pp. 488–503, 2013.
- [11] A. R. Pritchett and K. M. Feigh, "Simulating first-principles models of situated human performance," in *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. Miami Beach, FL: IEEE, 2011.
- [12] A. Pritchett, S. Y. Kim, S. K. Kannan, and K. M. Feigh, "Simulating situated work," in *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. IEEE, 2011, pp. 66–73.
- [13] J. Rushby, "The Needham-Schroeder protocol in SAL," *CSL Technical Note*, 2003.
- [14] E. J. Bass, K. M. Feigh, E. Gunter, and J. Rushby, "Formal modeling and analysis for interactive hybrid systems," in *4th International Workshop on Formal Methods for Interactive Systems*, Limerick, Ireland, June 2011.