

Aviation Safety: Modeling and Analyzing Complex Interactions between Humans and Automated Systems

Neha Rungta, Guillaume Brat, William J. Clancey, Charlotte Linde, Franco Raimondi[†], Chin Seah, Michael Shafto

NASA Ames Research Center, Moffett Field, CA 94035, USA

{neha.s.rungta,guillaume.p.brat,william.j.clancey,charlotte.linde,chin.h.seah,mike.shafto}@nasa.gov

[†]Middlesex University, London, UK, f.raimondi@mdx.ac.uk

ABSTRACT

The on-going transformation from the current US Air Traffic System (ATS) to the Next Generation Air Traffic System (NextGen) will force the introduction of new automated systems and most likely will cause automation to migrate from ground to air. This will yield new function allocations between humans and automation and therefore change the roles and responsibilities in the ATS. Yet, safety in NextGen is required to be at least as good as in the current system. We therefore need techniques to evaluate the safety of the interactions between humans and automation. We think that current human factor studies and simulation-based techniques will fall short in front of the ATS complexity, and that we need to add more automated techniques to simulations, such as model checking, which offers exhaustive coverage of the non-deterministic behaviors in nominal and off-nominal scenarios. In this work, we present a verification approach based both on simulations and on model checking for evaluating the roles and responsibilities of humans and automation. Models are created using Brahms (a multi-agent framework) and we show that the traditional Brahms simulations can be integrated with automated exploration techniques based on model checking, thus offering a complete exploration of the behavioral space of the scenario. Our formal analysis supports the notion of beliefs and probabilities to reason about human behavior. We demonstrate the technique with the Überlingen accident since it exemplifies authority problems when receiving conflicting advices from human and automated systems.

Author Keywords

Aviation Safety, Authority and Autonomy, Brahms, Verification, Überlingen collision.

ACM Classification Keywords

H.1.2 Models and Principles: User/Machine Systems; D.2.4 Software/Program Verification: Model Checking

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ATACCS2013, May 28-30 2013, Naples, Italy.

General Terms

Human Factors; Verification.

INTRODUCTION

Over the past few years, the US has embarked in a transformation of the Air Transportation System (ATS) to address the expected increase of air traffic in the US. The original prediction is that the traffic in 2025 will be between two and three times greater than the current traffic. There is no consensus on the actual size, but everybody agrees that the US needs to modernize the ATS and to implement NexGen (Next Generation Air Transportation System) to accommodate the traffic increase over the next 15 years; Europe is going through a similar effort with SESAR. An important goal of NexGen is to increase efficiency without compromising safety. The implementation of NexGen, described in the Integrated Working Plan (IWP), will see the introduction of new automated systems (e.g., ADS-B, GPS-based navigation) and new air traffic paradigms (e.g., 4D trajectory), which will cause some Air Traffic Management (ATM) functions to migrate from ground to on-board, and possibly vice versa. The new automation will cause a change in function allocation as well as a change in roles and responsibilities for air traffic controllers and pilots. As a consequence, it poses new challenges in assessing the safety of the overall system. This is the focus of our work.

The US National Airspace System (NAS) is currently quite safe and accidents are at a record low. NexGen needs to provide at least the same, if not a better, level of safety. The NexGen IWP has a requirement (R-1440) that calls for new and improved verification and validation (V&V) techniques for complex systems. This is often understood as applying only to the software systems that will be used in NexGen. However, we also need to recognize that NexGen is a complex system in which humans and autonomy (or automation: we do not need to make a distinction in this work) are interacting in quite subtle ways. Therefore, we also need new safety evaluation techniques to verify and validate the interactions of humans and autonomy in the complex system that is NexGen. Moreover, there is a large consensus that the earlier in the lifecycle this V&V is done, the easier it is to detect and fix errors [5]. In our work, we are focusing on developing methods for evaluating early, in the design phase, models of complex interactions in which there are multiple, different, simultaneous, situation-dependent assignments of

authority and autonomy (A&A) among humans and automation. In order to ensure safety in NexGen, there is a need for well-defined formalizations of procedures, possible actions of the actors involved, and the consequences of the actions. The formalization should facilitate the analysis of various interactions between the actors; the analysis can either be simulations or formal search-based techniques such as model checking. We specifically need to develop methods that allow us to articulate the authority bounds (limits) and behavior in terms of ownership: who has authority in any situation and how it may affect safety. The definitions that we adopt simplified definitions for authority and autonomy in this work are as follows:

- *Authority* refers to having the right, or power, to exercise controls or issue air traffic commands that impact the position, velocity, and/or attitude of aircraft during operations.
- *Autonomy* (or automation) refers to a function or system that can operate independently of pilot or air traffic controller intervention.

The ATS, especially with future NexGen concepts of operation, is a complex system involving dynamic interactions among multiple actors that are largely governed through formal assignment of roles and responsibilities. These A&A assignments are made at the design level, but are executed at the operational level according to each actor's view of its roles and responsibilities. Operationally, the system continuously adjusts for shortcomings in the assignment of authority and autonomy, for shortcomings in the capacity of actors to perform their assigned roles and responsibilities, and to optimize various performance factors such as capacity, environmental impact, and safety. This suggests that system safety should be derived not only from a predictable execution of assigned roles and responsibilities but also from checks and balances to ensure that the system operates as designed in the face of failures, disturbances and degradations. The ability of the system to operate in off-nominal conditions as a result of the checks and balances extent in it provides resilience, a critical characteristic for system safety.

Assessing safety in human-automation systems can be done using several techniques. Historically, human-in-the-loop studies have been the most prominent ones [4, 20]. They are quite costly to perform (they are real-time studies in which humans interact with ATS simulations), somewhat limited in scale (it is difficult to pull many controllers into a study) and often incomplete in the sense that they can explore only a restrictive set of behaviors. A few user interface rules have been extracted from these studies and can be used as building blocks to design the system interfaces. However, these rules fall short in helping in the context of a highly dynamic and complex system such as the ATS. Another way of analyzing human-machine systems is to create models for the humans (often based on the procedures that need to be performed by the humans) and run simulations [15, 18, 21] with the shortcoming that simulations can only examine a restricted set of behaviors. There has also been a growing interest and research in using formal methods for assessing

safety in human-automation systems, particularly in the aviation domain. They have the potential of exploring all possible behaviors given an sufficiently complex model for the human and the systems. Early examples of the use of model checking for analyzing human-machine interactions are described in [8, 13, 22]. More recent examples try to bridge the gap between simulations and the use of model checking [3, 6, 7]. The analysis method is model checking but the representation of the problem (i.e., models for the human and the automation) uses simulation languages instead of fairly simple finite state models. These techniques can even be expanded to the design and the verification of aerospace systems [2]. Our work falls in the category of using both simulation languages and formal methods. Our innovation resides in using a simulation language defined for representing multi-agent systems, which is what the ATS really is: a complex system of interacting agents some of which are humans and some of which are automated systems. But we also integrate the simulation language with formal verification techniques based on model checking.

Concretely, we model systems in the Brahms multi-agent framework [11, 24]. Brahms is a multi-agent simulation system in which people, tools, facilities, vehicles, and geography are modeled explicitly. The air transportation system is modeled as a collection of distributed, interactive subsystems such as airports, air-traffic control towers and personnel, aircraft, automated flight systems and air-traffic tools, instruments, and flight crew. Each subsystem, whether a person or a tool such as the radar, is modeled independently with properties and contextual behaviors. Brahms facilitates modeling various configurable realistic scenarios that allows the analysis of the airspace in various conditions and reassignment of roles and responsibilities among human and automation. We then apply formal methods to the proposed concepts and configurations early in the development process to identify promising candidates for safe solutions, as well as find design problems when they are easier to fix. This combination of modeling and formal methods will increase assurance of safety and motivate adoption of advanced automation and associated operations protocols. To motivate our approach we present a generalized air transportation system model based on the Überlingen collision.

The rest of the paper is organized as follows: to motivate our work we first describe the conditions that led to the Überlingen collision. Then, we describe how humans and automation, as well as their interactions, are modeled. Finally, we then present simulation results and a description the verification framework, and discuss related work.

ÜBERLINGEN COLLISION OVERVIEW

The Überlingen accident, [1], involving the (automated) Traffic Collision Avoidance System (TCAS), is viewed as a very good representative example illustrating the problem of authority versus autonomy (A&A) [10]. The Überlingen collision is a paradigmatic example of A&A conflicts. In particular, TCAS has the ability to reconfigure the pilot and air traffic control center (ATCC) relationship, taking authority from the air traffic control officer (ATCO) and instructing the pilot.

TCAS

TCAS is an onboard aircraft system that uses radar transponder signals to operate independently of ground-based equipment to provide advice to the pilot about conflicting aircraft that are equipped with the same transponder/TCAS equipment. The history of TCAS dates at least to the late 1950s. Motivated by a number of mid-air collisions over three decades, the United States Federal Aviation Administration (FAA) initiated the TCAS program in 1981. The system in use over Überlingen in 2002 was TCAS II v.7, which had been installed by US carriers since 1994:

TCAS II issues the following types of aural annunciations:

- Traffic advisory (TA)
- Resolution advisory (RA)
- Clear of conflict

When a TA is issued, pilots are instructed to initiate a visual search, if possible, for the traffic causing the TA. In the cases when the traffic can be visually acquired, pilots are instructed to maintain visual separation from the traffic. When an RA is issued, pilots are expected to respond immediately to the RA unless doing so would jeopardize the safe operation of the flight. The separation timing, called TAU, provides the TA alert at about 48 seconds and the RA at 35 seconds prior to a predicted collision.

Überlingen Collision Narrative

On July 1 2002, a midair collision between a Tupolev Tu-154M passenger jet travelling from Moscow to Barcelona, and a Boeing 757-23APF DHL cargo jet manned by two pilots, travelling from Bergamo to Brussels, occurred at 23:35 UTC over the town of Überlingen in southern Germany. The two flights were on a collision course. TCAS issued first a Traffic Advisory (TA) and then a Resolution Advisory (RA) for each plane. Just before TCAS RA to the Tupolev to climb, the air traffic controller in charge of the sector issued a command to descend, which the crew obeyed. Since TCAS had issued a Resolution Advisory to the Boeing crew to descend that they immediately followed, both planes were descending when they collided.

The decision of the Tupolev crew to follow the ATC's instructions rather than TCAS was the immediate cause of the accident. The regulations for the use of TCAS state that in the case of conflicting instructions from TCAS and ATCO, the pilot should follow the TCAS instructions. The conflict in the Überlingen scenario represents the conflict between the authority of automated systems (TCAS) and people (crews and ATC), as well as their autonomy (freedom to act independently). The reason this conflict came into being is because the loss of separation between the two planes was not detected or corrected by the ATCO. The loss of separation between airplanes are frequent occurrences; it is part of the normal work of air traffic control to detect and correct them accordingly.

There were a set of complex systemic problems at the Zurich air traffic control station that caused the ATCO to miss detecting the loss of separation between the two planes. Although two controllers were supposed to be on duty, one of

the two was resting in the lounge: a common and accepted practice during the lower workload portion of night shift. On this particular evening, a scheduled maintenance procedure was being carried out on the main radar system, which meant that the controller had to use a less capable air traffic tracking system. The maintenance work also disconnected the phone system, which made it impossible for other air traffic control centers in the area to alert the Zurich controller to the problem. Finally, the controllers workload was increased by a late arriving plane. An A320 that was landing in Friedrichshafen required the ATCO's attention, who then failed to notice the potential separation infringement of the two planes.

The Überlingen collision proves that methods used for certifying TCAS II v7.0 did not adequately consider human-automation interactions. In particular, the certification method treated TCAS as if it were flight system automation, that is, a system that automatically controls the flight of the aircraft. Instead, TCAS is a system that tells pilot how to maneuver the aircraft, an instruction that implicitly removes and/or overrides the ATCs authority. Worldwide deployment of TCAS II v7.1 was still in process in 2012, a decade after the Überlingen collision.

MODELING THE ÜBERLINGEN WORK SYSTEM

Overview of Brahms

Brahms is a full-fledged multi-agent, rule-based, activity programming language. It is based on a theory of work practice and situated cognition [11, 24]. The Brahms language allows for the representation of situated activities of agents in a geographical model of the world. Situated activities are actions performed by the agent in some physical and social context for a specified period of time [9]. The execution of actions is constrained (a) locally: by the reasoning capabilities of an agent and (b) globally by the agents beliefs of the external world, such as where the agent is located, the state of the world at that location and elsewhere, located artifacts, activities of other agents, and communication with other agents or artifacts. The objective of Brahms is to represent the interaction between people, off-task behaviors, multi-tasking, interrupted and resumed activities, informal interactions and knowledge, while being located in some environment representative of the real world.

The Brahms agent language can also be used to develop executable software agents that are based on models of situated behavior. This allows for the development of intelligent agents that can act and react to specific situations that occur during its execution, and that have been modeled as the agent's activity-behavior.

At each clock tick the Brahms simulation engine inspects the model to update the state of the world, which includes all of the agents and all of the objects in the simulated world. Agents and objects have states (factual properties) and may have capabilities to model the world (e.g., a radar's display is modeled as beliefs, which are representations of the state of the aircraft). Agents and objects communicate with each other; the communications can represent verbal speech, reading, writing, etc. and may involve devices such as telephones,

be working alone in the ATCC, while in another configuration, two controllers would be present in the ATCC. Initial beliefs of an agent might be broad preferences affecting behavior (e.g., TCAS should overrule the ATC), thus initial beliefs can be used as switches to easily specify alternative configurations of interest. Alternative configurations are conventionally called scenarios. Thus for example, a scenario might be a variation of the Überlingen collision in which two aircraft have inter-route flight times that put them on an intersecting path over Überlingen; the only other flight is a late arriving flight for Friedrichshafen and maintenance degrades the radar, but the telephones are operative.

In general, a model is designed by the model builder with sufficient flexibility to allow investigating scenarios of interest. The set of causal factors of interest (e.g., use of control strips when approving aircraft altitude changes, availability of telephones) constitute states of the world and behaviors that can be configured through initial facts and beliefs. The initial settings define a space of scenarios. Using Brahms to evaluate designs within this space, while using formal methods to help modelers understand its boundaries so they can refine the model to explore alternative scenarios, constitutes the main research objective of this work.

The simulation engine determines the state of a modeled object (e.g., aircraft). It determines the state of its facts and beliefs. Some objects are not physical things in the world, but rather conceptual entities, called conceptual classes in the Brahms language. These represent processes, a set of people, physical objects, and locations (e.g., flights), and institutional systems (e.g., airlines) that people know about and refer to when organizing their work activities.

High-level Structure of the Brahms Überlingen model

An overview of the agents, objects, classes in the Brahms Überlingen model are shown in Figure. 1. All of the systems that are mentioned in the BFU Report, [1], and play a role in accident have been modeled; a partial list follows:

1. Agents:
 - (a) Pilots in each aircraft
 - (b) Two ATCOs at Zurich
2. Geography:
 - (a) Airports: Moscow, Bergamo, Barcelona, Brussels
 - (b) Control Centers at Zurich and Karlsruhe that includes layout of physical workstations
 - (c) Aircraft interior layout
3. Objects:
 - (a) Aircraft: DHL, BTC, AEF, and other aircraft in the sector during the simulated time period (flights are conceptual objects associated with these).
 - (b) Flight Management Computer (FMC) with Cruise & Standard Terminal Arrival Route (STAR) modes for DHL & BTC

- (c) Control center workstations including radio frequencies and sectors.

4. Activities:

- (a) Flight Take-off Phase: Clock in ATCC announcing time for departure ATCO communicates departure approval; FMC guides with Standard Instrument Departure; Pilot activities and communications.
- (b) Flight Cruise Phase: FMC flying in auto-pilot mode using flight plan; Pilot activities and communications.
- (c) Flight Phase: Pilot activities and communications; ATCOs handoff and accept flights.
- (d) ATCOs handoff and accept flights
- (e) Flight Landing Phase: Pilot requests permission to land and ATCO communicates approval; FMC guides with Standard Terminal Arrival Route; Pilot activities and communications.

Key Subsystems and Conditions

The following key subsystems and conditions are modeled in the Brahms Überlingen model:

1. Interactions among Pilot, Flight Systems, and Aircraft for climb and cruise with European geography for one plane, the DHL flight plan.
2. BTC flight, flight plan (two versions: on-time and delayed with collision) and geography – this is independent of ATCO actions, to confirm that simulation reproduces collision with flight paths actually flown.
3. Radar Systems and Displays with ATCOs, located in Control Centers, monitoring when flights are entering and exiting each European flight sector in flight plans.
4. Handover interactions between Pilot and ATCOs for each flight phase.
5. Two ATCOs in Zurich, Radar Planner (RP), and AR-FARadar Executive (RE), assigned to two workstations (RE has nothing to do under these conditions).
6. Add TCAS with capability to detect separation violations, generate Traffic Advisory (TA) and Resolution Advisory (RA). DHL and BTC are delayed (on collision course, which tests TCAS)
7. Pilots follow TCAS instructions
8. ATCO may intervene prior to alert depending on when ATCO notices conflict in Radar Displays since ATCO is busy communicating with other flights, moving between workstations, and trying to contact Friedrichshafen control tower on the phone.
9. AEF flight and flight plan so Zurich ARFA RE performs landing handoff to Friedrichshafen controller.
10. Third plane, the AEF flight, arrives late, requiring ATCO communications and handoff to Friedrichshafen: (a) Handled by ATCO in Zurich at right workstation (ARFA sector) and not left East and South sector workstation. (b)

Phone communications for handovers, (c) Methods used by ATCO when phone contact does not work:

- (a) Ask Controller Assistant (CA) to get another number (pass-nr); requires about 3 minutes for CA to return
 - (b) After pass-nr fails, discuss with CA other options about 30 sec
 - (c) When not busy handling other flights, try pass-nr again.
 - (d) When plane is at Top-Of-Descent waypoint, as specified in STAR, for landing at airport, within N nm of airport, method of last resort is to call pilots on radio and ask them to contact the tower directly
11. STCA added to ATCO workstations (modeling normal and fallback mode without optical alert). The ATCO responds to alert by advising Pilot to change flight level based on next flight segment of flight plan.
 12. Reduce to one Zurich ATCO which triggers the sequence of variations from the nominal situation; now Zurich ATCO must operate flights from two workstations.

Note that fig:key does not show the geography, facilities, and flights.

PROPERTIES OF INTEREST

The question that the analysis tries to answer, using both simulation and verification, is why under certain conditions, a collision is averted, while in others it is not? In the analysis we try to gauge how the temporal sensitivity and variability of the interactions among ATCO, TCAS, and the pilots impacts the potential loss of separation and collision of the planes. Concretely, the questions that we ask during the analysis are:

- Given that the arrival of the AEF flight is disrupting the ATCOs monitoring of the larger airspace (e.g., if it arrives sufficiently late, no collision occurs), what is the period (relative to the BTC and DHL flights paths) when AEF's arrival can cause collision?
- During this period, does a collision always occur or are there variations of how the AEF handoff occurs, such that sometimes the separation infringement is averted?
- Is there evidence that high-priority activities such as monitoring the sector are repeatedly interrupted or deferred, implying the ATCO is unable to cope with the workload?

SIMULATION OF THE ÜBERLINGEN SCENARIOS

The Brahms Überlingen Model defines a space of work systems (e.g., is STCA optical functioning? are there two ATCOs?) and events (e.g., the aircraft and flights). Every configuration of model, which involves configuring initial facts, beliefs, and agent/object relations, constitutes a scenario that can be simulated and will itself produce many different outcomes (chronology of events), because of non-deterministic timings of agent and object behaviors. The model was developed and tested with a variety of scenarios (e.g., varying additional flights in the sector; all subsystems are working properly). The Überlingen accident is of special interest, in

which systems are configured as they were at the time of the accident and the DHL and BTC planes are on intersecting routes.

Setting up the Simulation

The key events that occur during simulation are logged chronologically in a file that constitutes a readable trace of the interactions among the ATCO, pilots, and automated systems. The log includes information about the following: (a) ATCO-pilot interaction regarding a route change, including flight level and climb/descent instruction, (b) Separation violation events detected by TCAS, including TAU value, (c) Closest aircraft and separation detected by ATCO when monitoring radar, (d) STCA optical or aural alerts, including separation detected, (e) Agent movements (e.g., ATCO shifting between workstations), (f) Aircraft movements, including departure, entering and exiting sectors, waypoint arrival, landing, collision, airspeeds, vertical, etc., (g) Aircraft control changes (e.g., autopilot disengaged), (h) Radio calls, including communicated beliefs, (i) Phone calls that fail to complete.

Summary of Results

The outcome of ten simulation runs of Brahms Überlingen model configured for the *collision* scenario are shown Figure. 2. In the simulation runs 1, 2, and 3 shown in Figure. 2, the ATCO intervenes before TCAS TA, but planes have not separated sufficiently, TCAS will take BTCs descent into account, advising DHL to climb. In the simulation runs 4, 5, 7, 8, and 9, the ATCO intervenes between TA and RA. In these runs whether the planes collide depends on timing. As shown in Figure 2 two of the five runs results in a collision. Note that in our model a *collision* is defined as occurring when the vertical separation between the planes is less than a 100 feet. Finally, in the simulation runs 6 and 10, the ATCO intervenes about 10 seconds after TCAS RA—which BTC pilots ignore (or might be imagined as discussing for a long time)—BTC continues flying level while DHL descends, so they miss each other, separated by more than 600 ft at the crossing point. In other runs, we have also observed that ATCO intervenes so late, he actually takes the pilots' report about TCAS RA instructions into account.

When ATCO intervenes in the period between the TA and RA in runs 4, 5, 7, 8, and 9 collision is possible, as at Überlingen. That is, ATCO has to intervene before TA advising BTC descent for BTC to respond sufficiently for TCAS to advise DHL to climb. In runs 4 and 7, collision is narrowly averted because BTC begins to descend four or five seconds after the TCAS RA, which is sufficient for a narrow miss (just over 100 feet). In run 9 the BTC descent begins 5 seconds before the RA, hence the aircraft miss by more than 200 feet). Runs 5 and 8 (Figure 9-3) lead to collision because the TCAS RA and BTC AP disengage occur at the same time, as happened at Überlingen. Because the model uses the Überlingen descent tables to control the BTC and DHL aircraft during the emergency descent, simulation matches the paths of the aircraft at Überlingen guaranteeing a collision (within defined range of error). In both cases, TCAS didn't instruct DHL to climb because BTC was above DHL at that time and of course had not begun its descent.

Run #	Collide?	Explanation	ATCO-BTC	TCAS RA-DHL	ATCO relative TA/RA
1	No	TCAS detects BTC plane descending due to ATCO; so advises DHL to Climb.	Descend	Climb	Before
2	No	TCAS detects BTC plane descending due to ATCO, so advises DHL to Climb.	Descend	Climb	Before
3	No	TCAS detects BTC plane descending due to ATCO, so advises DHL to Climb. AEF flight arrives very late after TCAS TA.	Descend	Climb	Before
4	No	DHL TCAS Descend; BTC above. Planes crossed > 100 ft vertical separation	Descend	Descend	During
5	YES	DHL TCAS Descend; BTC above. BTC AP turned off at DHL RA. Planes crossed < 20 ft vertical separation	Descend	Descend	During
6	No	DHL TCAS Descend; BTC above. ATCO later than RA, so BTC level. Planes crossed > 600 ft vertical separation	Descend	Descend	After
7	No	DHL TCAS Descend; BTC above. DHL AP turned off 2 seconds before BTC. Planes crossed > 100 ft vertical separation	Descend	Descend	During
8	YES	DHL TCAS Descend; BTC above. BTC AP turned off at RA. Planes crossed < 50 feet vertical separation	Descend	Descend	During
9	No	DHL TCAS Descend; BTC above. Planes crossed > 200 ft vertical separation	Descend	Descend	During
10	No	DHL TCAS Descend; BTC above. ATCO later than RA, so BTC level. Planes crossed > 600 ft vertical separation	Descend	Descend	After

Figure 2. Outcomes of ten simulation runs of Überlingen scenario. Bold indicates greatest potential for collision (ATCO intervenes between TA and RA; both aircraft descending).

When ATCO intervenes after the RA, the BTC pilots in the simulations ignore the RA advice and continue level flight, which itself averts the collision—even though ATCO advises BTC to descend (which implies not considering that DHL is below them). We of course do not know what the BTC pilots would have done if ATCO had not intervened. With more than one pilot interpreting TCAS correctly, it appears possible the BTC would have climbed.

The final AEF hand-off (directing the pilots to contact the tower) always occurs in the simulation after the TCAS RA; at Überlingen it occurred prior to the TA. This discrepancy raises many questions about what variability is desirable. In the verification of the system we were able to find certain cases where the final AEF hand-off occurs before the TCAS TA and the planes still collide.

The simulation results for other configurations of the Brahm's Überlingen model are described in the technical report [10].

FORMAL VERIFICATION

We use verification techniques to systematically explore the various behaviors in collision scenario of the Brahm's Überlingen model configuration in addition to the simulation experiments.

Background

In [16] we present an extensible verification framework that takes as input a multi-agent system model and its semantics as input to some state space search engine (or a model checker). The search engine generates all possible behaviors

of the model with respect to its semantics. The generated behaviors of the model are then encoded as a reachability graph $G := \langle N, E \rangle$ where N is a set of nodes and E is a set of edges. This graph is automatically generated by the search engine. Each node $n \in N$ is labeled with the belief/facts values of the agents and objects. In the work in [16] we generate the reachability graph using the Java PathFinder bytecode analysis framework. An edge between the nodes represents the updates to beliefs/facts and is also labelled with probabilities. The reachable states generated by the JPF are mapped to the nodes in a reachability graph. The verification of safety properties and other reachability properties is performed on-the-fly as new states and transitions are generated in JPF. Additional verification activities can be performed on the reachability graph after all the JPF states have been generated.

Limitations The JPF-based MAS connector requires a complete implementation of the Brahm's semantics to generate the intermediate representation. The current implementation of the Brahm's semantics presented in [16] only supports a limited set of constructs. Furthermore, JPF is a stateful analysis engine that stores the generated model in memory. Capturing the state of all the agents and objects in Brahm's including their workframes and thoughtframes can lead to large memory requirements. Additionally, for large systems it is often intractable to generate and capture even just the intermediate representation in memory.

Stateless Brahm's Model Checking

To overcome the limitations just described, in this work we adopt a *stateless* model checking approach. *Stateless model checking* explores all possible behaviors of the program or model without storing the explored states in a visited set. The program or model is executed by a scheduler that tracks all the points of non-determinism in the program. The scheduler systematically explores all possible execution paths of the program obtained by the non-deterministic choices. Stateless model checking is particularly suited for exploring the state space of large models. In this work we instrument the Brahms simulator to perform stateless model checking. The instrumented code within the Brahms engine generates all possible paths (each with different combinations of activity durations) in depth-first ordering. Stateless model checkers like VeriSoft [14] do not in general store paths; however, in order to perform further analysis of the behaviors space the Brahms stateless model checker can store all the generated paths in a database.

Non-determinism in Brahms

There are two main points of non-determinism in Brahms models. The first point of non-determinism is due to durations of primitive activities. The different primitive activities in Brahms have a duration in seconds associated with them. The duration of the primitive activity can either be fixed or can vary based on certain attributes of the primitive activities. When the random attribute of a primitive activity is set to true the simulator randomly selects the primitive activity duration between the min and max durations specified for the activity. The second point of non-determinism arises from probabilistic updates to facts and beliefs of agents and objects. Updates to facts and beliefs are made using `conclude` statements in Brahms. An example of a `conclude` statement is: `conclude((Pilot.checkStall = false), bc:70, fc:70)`. This states that the belief and fact, `checkStall`, in the Pilot agent will be updated to false with a probability of 70%. Here `bc` represents belief certainty while `fc` represents fact certainty.

In the Überlingen model currently there are only deterministic updates to facts or beliefs. The updates to facts and beliefs are asserted with a 100% probability. Nevertheless, there is a large degree of non-determinism due to variations in activity durations. The difference in minimum and maximum duration ranges from 2 seconds to a few hundred seconds. This can potentially lead to a large number of timing differences between the various events. In future work we plan to extend the Brahms Überlingen model to support probabilistic variations in order to account for errors by humans and automated systems.

Behavior Space

The scheduler within the stateless Brahms model checker generates all possible paths through the different points of non-determinism in the Brahms model. Note that in describing the output of the Brahms stateless model checker we use the terms path and trace interchangeably. Intuitively, a path (or trace) generated by the Brahms stateless model checker is equivalent to the a single simulation run. More formally, a path or trace is a sequence of events executed by the sim-

ulator $\langle e_0, e_1, e_2, \dots, e_i \rangle$. Each event in the trace is a tuple, $\langle a, t, (u, val) \rangle$ where a is the actor id, t is the Brahms clock time, u is the fact or belief updated to the value val . For each trace we generate a sequence of nodes in the intermediate representation $n_{init}, n_0, n_1, n_2, \dots, n_i$. The initial node in the sequence, n_{init} is labeled with the initial values of belief/facts values for the various agents and objects. The event $e_0 := \langle a_0, t_0, (u_0, val_0) \rangle$ is applied to the initial node n_{init} where the value assigned to u_0 is updated to val_0 . Each event is applied in sequence to a node in the intermediate representation to generate $n_{init}, n_0, n_1, n_2, \dots, n_i$.

Summary of the Results

There are several activities in the Brahms Überlingen model with a specified range of minimum and maximum durations. Due to the size and complexity of the model, generating a single trace takes approximately 15 minutes. It would in all likelihood take a few weeks to generate all possible traces within the system. In order to mitigate this computational bottleneck, we scope the verification of the model. We non-deterministically explore the minimum, median, and maximum durations for each activity in the model. In the traces generated by the stateless Brahms model checker, approximately a third of the generated traces lead to a collision. If the collision was an undesired property (a fault) in the model, then the results of the model checking would indicate a very high error density. It is, however, important to note that the goal of the collision configuration in the Brahms Überlingen model was to faithfully recreate the conditions that led to the planes colliding. The verification results demonstrate that even with the timing variations a large number of paths (one-third of the generated paths) lead to the collision due to the fact that the ATCO was distracted with the AEF flight, the short-term collision avoidance system (STCA) which provides optical and audible alerts for the ATCO was under maintenance, and the fact that there was only one ATCO on duty.

We present an overview of the verification results for the two properties of interest described earlier: (a) how does the arrival of the AEF flight impact the ATCO's ability to monitor the large airspace and (b) does a collision always occur in this period? Some of the results described in the simulation also hold true for the traces generated during the verification. We were able to study other interesting aspects of the model with respect to the properties that were not observed in the simulation. In the simulation runs of the Überlingen model the final AEF handoff (directing the pilots to contact the tower) always occurs in the simulation after the TCAS RA; in the verification runs, however, the final AEF hand-off can occur before the TA is ever issued. Some of the cases observed are as follows:

1. The final AEF hand off occurs before the TA, the separation infringement is detected and resolved.
2. The final AEF hand off occurs before the TA, and the planes still *collide*. Note that this is a very interesting scenario because the Überlingen accident report states that the the final AEF hand off occurred before the TA for either of the planes.

The verification results indicate that while in some cases the AEF flight arrival can exacerbate the problem for the ATCO, it is not the only cause of the accident. From a wider systemic perspective, the separation violation did not occur at Überlingen only because of the arrival time of the AEF flight. Rather, the Skyguide company had tolerated a deviant form of SMOP during night operations: consequently nobody was carrying out the role of the supervisor in the ATC. Nobody was responsible for the system, particularly during the maintenance process. Otherwise ATCO would have been informed that STCA Optical alert was not functioning and that the backup phones had been disabled. We can encode the output of the Brahms Stateless model checker into a PRISM model, [17], and check various probabilistic properties of the system. The updates to the facts and beliefs represent the probabilistic updates to the system. Note that the output of the Brahms stateless model checker can be encoded as the intermediate representation in the work in [16].

The Überlingen collision scenario does not provide opportunity for sophisticated properties since a large number of paths lead to the collision of the planes. The model, however, lends itself to be extended to other general cases and scenarios present in the aviation domain. For example, most pilots in practice commonly ignore the TA alert issues by TCAS, but are trained to react immediately to an RA. Rather than being specified as initial configuration we can extend the model to support probabilistic updates that indicate whether or not the phones are down, whether the STCA is in maintenance, and the other ATCO officer is on a break.

DISCUSSION

Our overall goal is to model and analyze interactions between humans and automated systems, and apply this methodology to the safety analysis of NextGen. It is our conjecture that such an analysis needs to be done early in design before deploying any new automation. The problems that are related to safety which can be detected early in the design phase are easier to fix. In order to achieve this goal, we need to reason about how humans perform their tasks in conjunction with complex, thus hard to grasp in its entirety, automation. It led us to making the following choices.

To model the interactions between humans and automation we chose the Brahms modeling language. Brahms has the ability to reason about agents and objects that can represent humans as well as automated systems. The agents can have varying levels of intelligence which provides us the flexibility to model agents at varying granularity. The simulation of agent behavior can range from rational procedure following to simulating how people actually doing their work, i.e., their practice, or simulating reactive behavior that is fragmented, unfocused, incomplete, etc. We can encode non-deterministic choices in the model and even assign probabilities to these choices. We can also express the notion of belief, which is quite important when a human interacts with a complex system. For example, the pilot in charge during the Air France 447 accident described in [16] had wrong assumptions about the pitch of the plane and being able to model his belief as to the state of the system is important. Brahms also gives us the

added benefit of being able to model precisely a working environment (e.g., a controller console is two yards away from another one, which implies some time is needed to switch from one to the other). Early in the design phase, details, including those about work settings, are not necessarily known. However, it is advantageous to have this feature when one wants to refine the analysis as one gets closer to deployment at particular locations. Quite often FAA ground systems require adaptations when they are fielded at new locations. Using Brahms' capabilities, one can tune a generic model to the details of a particular location and verify that no new safety issues can appear.

From an analysis point of view, we are taking a pragmatic approach, adopting both simulations and model checking techniques. In this work, we experimented with generating the behaviors using a stateless model checker. The reachability graph can also be generated by the JPF model checker as described in [16] where currently a subset of the Brahms semantics are implemented as a Java library. After completing the implementation of the Brahms semantics in JPF we can leverage the several extensions of JPF to facilitate the scalability of the analysis. Another important criterion is to provide the ability to reason about beliefs and probabilistic behaviors. As described in the verification framework of [16], we can encode the reachability graph into inputs for different model checkers such as PRISM, SPIN, and NuSMV. This allows us to leverage state of the art verification technologies and check properties related to probabilities, liveness, and beliefs.

With respect to the methodology, it is important to reconcile the need for details in an analysis with the fact of performing an analysis early in design when details are not necessarily known. Our first answer is based on using fairly generic models for controllers and pilots based on the current literature (which includes the body of work in human factor studies). These generic models can then be refined as we progress towards implementation, adaptation, and finally deployment. Scalability might still be an issue, and we will address it by using proper abstractions and, if possible, compositional verification techniques. Similarly, we will have to address the scalability of the models when it comes to analyzing larger parts of the National Airspace System (e.g., multiple airports, multiple sectors, many airplanes). Fortunately JPF is being extended with capabilities to address abstractions and compositional verification. So, we will have a good base from which to draw.

RELATED WORK

In addition to the approaches mentioned in the Introduction, there is a large body of work dealing with the verification of human-machine interactions and with the verification of avionic systems. The DO-178B titled *Software considerations in airborne systems and equipment certification* is the official guideline for certifying avionics software. Several model checking and formal verification techniques have been employed to verify avionic *software* in [19, 2] in accordance with the DO-178B. Recent work describes how changes in aircraft systems and in the air traffic system pose new challenges for certification, due to the increased interaction and

integration [23].

In [19] the authors present a framework that supports multiple input formalisms to *model* avionic software: these include MATLAB Simulink/Stateflow and SCADE. These formalisms are then translated into an intermediate representation using Lustre, a standard modelling language employed to model reactive systems with applications in avionics. Finally, Lustre models are translated to the input language of various model checkers, including NuSMV, PVS, and SAL. The key difference with the approach we describe for formal verification is that the translation is purely *syntactical*. In our work, instead, we do not translate the modelling language, but we operate at the level of the Brahms simulator. This allows us to consider the full semantics of Brahms, and not a subset of the language compatible with the verification tools. More importantly, we explicitly consider a hybrid system composed of software and humans, and we are able to reason about beliefs and probabilities, while the work in [19] is limited to temporal properties.

There is a vast literature to model human-machine interactions. Recently, Combefis et al. [12] have employed Java Pathfinder as a model checker to verify human-machine interactions. The modelling language is based on Statecharts but, as in the work of [19], this formalism does not allow us to reason about probabilities or beliefs. We refer to the references available in [12] for an overview of other similar approaches.

The work of Yasmeen and Gunter [25] deals with the verification of the behaviour of human operators to check the robustness of mixed systems. In this approach the authors employ concurrent game structures as the modelling language and translate the verification problem to a model checking instance using SPIN. As in the previous cases, our approach is different in that we do not perform syntactic translations and we reason explicitly about probabilities and beliefs. Additionally, we also provide a detailed and complex case study.

The Enhanced Operator Function Model (EOFM) is another modelling language developed to model and verify interactions between humans and automated systems [7]. Similarly to the other works described above, EOFM is translated into the input language of the model checker SAL to perform verification of properties encoded in linear temporal logic. The authors describe the application of their framework to the verification of a cruise control system for cars. The main limitation of this approach is that it currently supports single-operator systems only and, as in the case of [19] and [25], there is no support to reason about probabilities and beliefs.

Acknowledgements This work is supported by the System-Wide Safety and Assurance Technologies Project in the Aviation Safety Program of NASA's Aeronautics Research Mission Directorate. The views expressed in this paper are those of the authors and do not necessarily represent the views of NASA or the U.S. Government. Author names are alphabetically ordered following the first author. Sections of this paper describing Überlingen collision, Brahms framework, Brahms Überlingen model, and simulation scenarios

are adapted from [10]. We thank Ron van Hoof providing support for the Brahms simulation system.

REFERENCES

1. Report. 2004. bundesstelle fur flugunfalluntersuchung investigation report german federal bureau of aircraft accidents investigation. ax001-1-2, 2002.
[http://www.skybrary.aero/index.php/T154/_B752,_enroute,_Uberlingen_Germany,_2002_\(LOS_HF\)](http://www.skybrary.aero/index.php/T154/_B752,_enroute,_Uberlingen_Germany,_2002_(LOS_HF)).
2. Ait Ameur, Y., Boniol, F., and Wiels, V. Toward a wider use of formal methods for aerospace systems design and verification. *Int. J. Softw. Tools Technol. Transf.* 12, 1 (Jan. 2010), 1–7.
3. Bass, E. J., Bolton, M. L., Feigh, K. M., Griffith, D., Gunter, E. L., Mansky, W., and Rushby, J. M. Toward a multi-method approach to formalizing human-automation interaction and human-human communications. In *SMC, IEEE* (2011), 1817–1824.
4. Benson, K. C., Dean, G., Kuhlmann, G., Sperling, B., Pritchett, A. R., and Jacko, J. A. Experimental study of cockpit displays of traffic information for pilot self-spacing in congested airspace. In *CHI Extended Abstracts*, G. Cockton and P. Korhonen, Eds., ACM (2003), 842–843.
5. Boehm, B. *Software Engineering Economics*. Prentice Hall, 1981.
6. Bolton, M. L., and Bass, E. J. Enhanced operator function model: A generic human task behavior modeling language. In *SMC, IEEE* (2009), 2904–2911.
7. Bolton, M. L., Siminiceanu, R. I., and Bass, E. J. A systematic approach to model checking human-automation interaction using task analytic models. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 41, 5 (2011), 961–976.
8. Chakraborty, S., Jerraya, A., Baruah, S. K., and Fischmeister, S., Eds. *A formal framework for design and analysis of human-machine interaction*, IEEE (2011).
9. Clancey, W. J. Simulating activities: Relating motives, deliberation, and attentive coordination. *Cognitive Systems Research* 3, 3 (2002), 471–499.
10. Clancey, W. J., Linde, C., Seah, C., and Shafto, M. The Brahms Generalized Überlingen Model. *Technical Report, Publ. 2013-216508*. (2013).
11. Clancey, W. J., Sachs, P., Sierhuis, M., and Van Hoof, R. Brahms: Simulating practice for work systems design. *International Journal of Human-Computer Studies* 49, 6 (1998), 831–865.
12. Combéfis, S., Giannakopoulou, D., Pecheur, C., and Feary, M. A formal framework for design and analysis of human-machine interaction. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Anchorage, Alaska, USA, October 9-12, 2011*, IEEE (2011), 1801–1808.

13. Degani, A., and Heymann, M. Formal verification of human-automation interaction. *Human Factors* 44, 1 (2002), 28–43.
14. Godefroid, P. Model checking for programming languages using verisoft. In *Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ACM (1997), 174–186.
15. Göknur, S., Bolton, M. L., and Bass, E. J. Adding a motor control component to the operator function model expert system to investigate air traffic management concepts using simulation. In *SMC (1)*, IEEE (2004), 886–892.
16. Hunter, J., Raimondi, F., Rungta, N., and Stocker, R. A synergistic and extensible framework for multi-agent system verification. *To Appear, Proceedings of the AAMAS 2013 : Twelfth International Conference on Autonomous Agents and Multiagent Systems* (2013).
17. Kwiatkowska, M., Norman, G., and Parker, D. Prism: Probabilistic symbolic model checker. In *Computer Performance Evaluation: Modelling Techniques and Tools*. Springer, 2002, 200–204.
18. Lee, S., Pritchett, A. R., and Goldsman, D. Hybrid agent-based simulation for analyzing the national airspace system. In *Winter Simulation Conference* (2001), 1029–1036.
19. Miller, S. P., Whalen, M. W., and Cofer, D. D. Software model checking takes off. *Commun. ACM* 53, 2 (Feb. 2010), 58–64.
20. Pritchett, A. R., Fleming, E. S., Cleveland, W. P., Zoetrum, J. J., Popescu, V. M., and Thakkar, D. A. Pilot interaction with tcas and air traffic control. In *ATACSS, W. Y. Ochieng and F. J. Saez, Eds., IRIT Press Toulouse, France / ACM DL* (2012), 117–126.
21. Pritchett, A. R., and Wieland, F. Preface to special issue on air transportation. *Simulation* 83, 5 (2007), 371–372.
22. Rushby, J. Using model checking to help discover mode confusions and other automation surprises. *Reliability Engineering and System Safety* 75, 2 (2002), 167–177.
23. Rushby, J. M. New challenges in certification for aircraft software. In *EMSOFT, S. Chakraborty, A. Jerraya, S. K. Baruah, and S. Fischmeister, Eds., ACM* (2011), 211–218.
24. Sierhuis, M. *Modeling and Simulating Work Practice. BRAHMS: a multiagent modeling and simulation language for work system analysis and design*. PhD thesis, Social Science and Informatics (SWI), University of Amsterdam, SIKS Dissertation Series No. 2001-10, Amsterdam, The Netherlands, 2001.
25. Yasmeen, A., and Gunter, E. L. Automated framework for formal operator task analysis. In *Proceedings of the 20th International Symposium on Software Testing and Analysis, ISSTA 2011, Toronto, ON, Canada, July 17-21, 2011, M. B. Dwyer and F. Tip, Eds., ACM* (2011), 78–88.