JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 12, No. 7, July 2015

Model Checking Human–Human Communication Protocols Using Task Models and Miscommunication Generation

Matthew L. Bolton*
University at Buffalo, State University of New York, Buffalo, New York 14260

DOI: 10.2514/1.I010276

Human–human communication is critical to safe operations in air transportation systems. For example, airlines develop and train pilots to use communication protocols designed to ensure that verbally communicated air traffic clearances are correctly executed by the pilots. Given the safety criticality of such interactions, these protocols should be designed to be robust to miscommunication. However, designers may not anticipate all of the different ways that miscommunication can occur. Thus, communication protocols can fail. This paper presents a method for evaluating human–human communication protocols using the enhanced operator function model with communications, which is a task analytic modeling formalism that can be used in model-checking formal verification analyses. In particular, a novel means of generating miscommunications from normative human–human communication protocols, instantiated as enhanced operator function models with communications, is introduced. An air transportation example is used to illustrate the power of the approach, where a protocol is iteratively evaluated and improved to make it robust for multiple miscommunications. Different versions of the application protocol are used to assess the scalability of the method. Results are discussed, and avenues of future research are explored.

I. Introduction

HUMAN–HUMAN communication is critical to the safe operation of many complex systems. Until the widespread adoption of controller/pilot data link communication, voice communications will remain the primary means air traffic controllers have for communicating clearances to pilots [1,2]. To facilitate safe and consistent operations, these communications must follow protocols [3]. For example, pilot and controller communications will often employ readbacks and other confirmation behaviors to help ensure that the intended information has been properly conveyed [3–5]. Despite these precautions, miscommunications still occur and can affect the safety and efficiency of the airspace system [4–7]. For example, in an analysis of more than 12,000 air traffic incident reports, Billings and Reynard [8] found that 73% had information transfer problems. Similarly, the Flight Safety Foundation's Approach-and-Landing Accident Reduction Task Force identified that inadequate communication was a causal factor in 33% of the accidents and serious incidents for approach and landing worldwide from 1984 to 1997 [1].

To prevent miscommunications, institutions will try to enhance human–human communication protocols. However, even in relatively simple protocols, it can be difficult to anticipate all of the possible human–human interactions, and thus design protocols guaranteed to be robust to miscommunications. There is a clear need for tools and techniques capable of providing such guarantees. This paper presents such a technique. The presented approach uses formal methods (tools for mathematically modeling and proving properties about systems) [9] with task analytic and human communication modeling. The remainder of this paper covers the relevant background on formal methods and their use to evaluate communication protocols necessary to contextualize and motivate the presented material. Then, the research objectives of this work are described, followed by a description of the developed method. To demonstrate the power of the method, it is then applied to a realistic aerospace problem. Finally, we discuss our results and future research possibilities.

II. Background

A. Formal Methods

Formal methods are tools and techniques for the modeling, specification, and verification of systems [9]. Modeling involves describing the behavior of a target system using a robust mathematical formalism. Specifications mathematically assert desirable properties about the system. Verification mathematically proves that the model adheres to the specification properties. Model checking is used in the presented work. Model checking is a software tool that can automatically prove if a system model (usually modeled as a finite state transition system) adheres to specification properties (usually represented in temporal logic) using exhaustive search algorithms [10].

Formal methods have been used successfully to find and help correct bugs in a number of computer software and hardware systems. Further, there is a long tradition of formal methods being used to describe and formally verify machine communication protocols [10–15], including the injection of communication faults or errors into the models of the protocols [16,17].

B. Formal Methods for Human–Human Communication Behavior

Although human–human communication protocols could be modeled using traditional formal methods approaches, human–human communication, which can include verbal statements, gestures, and related actions, is different from machine communication. In this context, human communications are actions [18] that occur as part of the participants' larger tasks. For analysis and engineering purposes, tasks are usually documented as part of a task analysis [19] and are designed to capture the goal-directed normative behaviors humans use to accomplish goals with

The work documented here is an extended version of results originally presented at the Fifth NASA Formal Methods Symposium in May 2013 at the NASA Ames Research Center titled "Evaluating Human–Human Communication Protocols with Miscommunication Generation and Model Checking." This new paper extends the work from original proceedings by iteratively applying the described miscommunication generation technique and the EOFMC-supported formal verification analyses to multiple versions of the original protocol to improve protocol robustness. It also reports scalability benchmarks and features an expanded introduction and discussion.

Received 18 April 2014; revision received 12 October 2014; accepted for publication 26 November 2014; published online 30 January 2015.

*Assistant Professor, Department of Industrial and System Engineering; mbolton@buffalo.edu.

the system. There are a number of different ways to represent task models, but hierarchal models are the dominant paradigm. In such models, tasks are represented as goal-directed activities that can decompose into lower-level activities (subgoals) and, at the lowest level, actions. Conditions on activities indicate when they are relevant and what they should accomplish, while modifiers between activities and actions control how they should be performed relative to each other. Task models can be used for a number of different analysis and engineering purposes [19], and are thus common in human factors engineering.

Few researchers have used formal methods to evaluate human–human communication protocols. Hörl and Aichernig [20,21] developed a formal model of the system that pilots and air traffic controllers use to communicate. However, rather than perform formal verification, they used automated test case generation to develop scenarios to guide human subject testing. Others have investigated how task analytic models can be incorporated into formal models and evaluated with formal verification (see [22] for a review). Only [23–24] treated human–human communication as actions within a larger set of coordinated communication and task-relevant activities. Paternò et al. [23] extended Concur Task Trees (a task modeling formalism) [25] to allow for the modeling of human–human coordination and communication. In this, Paternò et al. modeled human–human communications as actions with different modalities (synchronous or asynchronous, point-to-point, or broadcast) and used them to represent pilot and air traffic control radio communications during runway operations using different shared task representations. With the enhanced operator function model with communications (EOFMC), Bass et al. [24] extended the enhanced operator function model (EOFM) [26] to support modeling human–human communication and coordination as shared task structures between human operators. If a task goal is only associated with a given human operator, he or she can have separate, unshared tasks. This allows the activities associated with a given communication protocol to be contained in a separate task structure that can be analyzed on its own or with other modeled tasks. However, neither of the approaches presented in [23–25] have investigated how miscommunications could result in the failure of human–human communication protocols.

III. Objectives

Methods are needed to support the formal evaluation of human–human communication protocols with the inclusion of potential miscommunications. This paper describes a novel approach (first introduced in [27]) developed to automatically generating miscommunications as part of human–human communication protocols represented in task models. It shows that this can be used to formally prove that a given human–human communication protocol, represented in the context of a task analytic model, will or will not always accomplish its goals even with potentially unanticipated miscommunications. Further, the work is extended from [27] to demonstrate that the method can be used to explore different designs to improve the robustness of a protocol to miscommunications. The examined protocol is also used to evaluate how the presented methods scales (another contribution beyond what was reported in [27]). In the following sections, the method and its implementation are described. First, EOFMC, the task analytic modeling formalism used for modeling protocols, is described along with the process that is used to translate instantiated EOFMCs into the input language of a model checker. The novel miscommunication generation method is then introduced, and its implementation within EOFMC is explained. An air traffic control application is presented to illustrate how the method can discover problems with safety-critical human–human communication protocols. Using the method, it is shown how analysts can explore different communication protocols to find one that accomplishes their safety goals. The protocols developed for the application are also used to benchmark the scalability of the miscommunication generation process. Ultimately, the results of this effort are discussed and avenues of future research are outlined.

IV. Method

The method in Fig. 1 was extended from [28,29] to allow an analyst to evaluate whether or not a human–human communication protocol will accomplish its goals for up to a specified number of miscommunications. An analyst starts by creating a human–human communication protocol in a task analytic modeling formalism. The result is run through a translation process that produces a representation of the protocol in the input language of a model checker. This version of the model includes the maximum number of miscommunications that the analyst wants in the verification process. The analyst also creates a specification that asserts desirable properties about the communication protocol in a formal specification language such as a temporal logic. Model checking performs formal verification, checking whether the formal model of the communication protocol adheres to the specification [10]. Model checking produces a verification report, confirming that the model adheres either to the specification or a counterexample, which illustrates how the specification was violated.

A. Human–Human Communication Protocol Modeling

The EOFMC [24] (an extension of the EOFM [26,30,31]) is used to model human–human communication protocols. The EOFMC was employed for several reasons. First, the EOFMC is capable of representing multiple human operators and human–human communication as part of a larger task model. As part of this, it allows communication protocols to be modeled as shared task structures on their own or with other tasks. Second, EOFMC has a formal semantics [24] that allow tasks instantiated in it to be automatically translated into formal languages that can be used in model-checking analyses. Finally, the EOFM supports two different erroneous behavior generation methods [32,33] suggesting that the EOFMC would be readily adaptable to miscommunication generation.

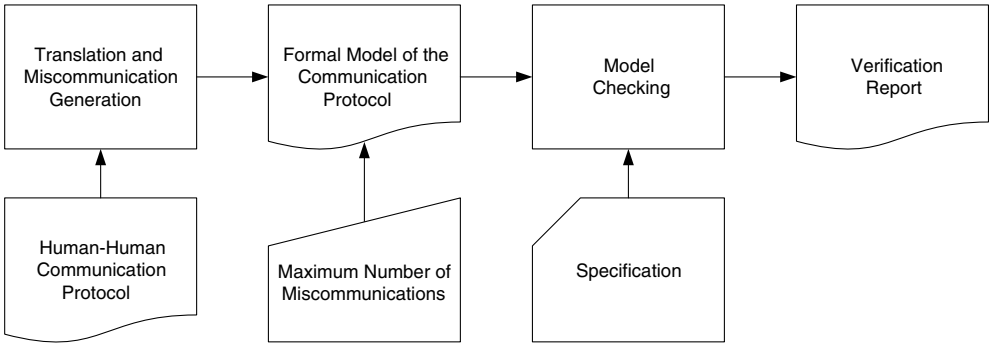

Fig. 1 Human–human communication protocol analysis method.

The EOFMC represents groups of human operators engaging in shared activities as an input/output system. Inputs may come from a human interface, environment, and/or mission goals. Output variables are human actions. The operators' task models describe how human actions may be generated and how the values of local variables change based on input and local variables (representing perceptual or cognitive processing, task behavior, and inner group coordination and communication). All variables are defined in terms of constants, user-defined types, and basic types.

Tasks in an instantiated EOFMC[†] are represented as a hierarchy of goal-directed activities that ultimately decompose into atomic actions. Each task descends from a top-level activity (there can be many tasks in a given instantiated EOFMC). Tasks can either belong to one human operator or they can be shared between human operators. A shared task is associated with two or more associates, and a subset of associates for the general task are identified for each activity. Thus, it is explicit which human operators are participating in which activity.

Activities can have preconditions, repeat conditions, and completion conditions. These are represented by Boolean expressions written in terms of input, output, and local variables, as well as constants. They specify what must be true before an activity can execute (precondition), when it can execute again (repeat condition), and what is true when it has completed execution (completion condition).

Actions occur at the bottom of the task hierarchy. They can assume several forms:

1) They can be observable, singular ways the human operator can interact with the environment.

2) They can represent a cognitive or perceptual act, where a value is assigned to a local variable.

3) They can represent human–human communications, where a communicator performs a communication action and the information conveyed (which will have a defined type) is stored in recipient local variables.

A decomposition operator specifies the temporal relationships between and the cardinality of the decomposed activities or actions (when they can execute relative to each other and how many can execute). The EOFMC supports 10 different decomposition operators (see [24]). Herein, only the following are used:

1) The *and_par* operator indicates that all activities or actions in the decomposition must execute (in any order) and their execution can overlap.

2) The *ord* operator indicates that all activities or actions in the decomposition must execute in order.

3) The *com* operator indicates that all of the actions in a decomposition must execute synchronously, where one human operator must perform a communication action and at least one other human operator must be the recipient of that communication.

The structure of an instantiated EOFMC can be represented visually as a treelike graph, where actions are depicted by sharp-edged rectangles and activities are depicted by round-edged rectangles. Conditions are connected to the activity they modify: a precondition is represented by a downward-pointing triangle connected to the left side of the activity; a *CompletionCondition* is presented as an upward-pointing triangle connected to the right of the activity; and a *RepeatCondition* is conveyed as a recursive arrow attached to the top of the activity. Decompositions are arrows, labeled with the decomposition operator, extending below an activity that point to a large, rounded rectangle with the decomposed activities or actions.

By exploiting the shared activity and communication action feature of the EOFMC, human–human communication protocols can be modeled as shared task activities. Human communication actions can represent human–human communication. However, other actions can model the way that the human operator interacts with other elements of the work environment. Thus, a human–human communication protocol can represent the human–human communication procedure and the human operator responses.

B. EOFMC Formal Semantics and Translation

The EOFMC has formal semantics that specify how an instantiated EOFMC model executes. Each activity or action has one of three execution states: *Ready* (waiting to execute), *Executing*, and *Done*. An activity or action transitions between states based on its current state; its start condition (*StartCondition*: when it can start executing based on the state of its immediate parent, its parent's decomposition operator, and the execution state of its siblings); its end condition (*EndCondition*: when it can stop executing based on the state of its immediate children in the hierarchy and its decomposition operators); its reset condition (*Reset*: when it can revert to *Ready* based on the execution state of its parents); and, for an activity, its strategic knowledge (the *Precondition*, *RepeatCondition*, and *CompletionCondition*). See [26] for more details.

The instantiated EOFMC task models can be translated into the language of the Symbolic Analysis Laboratory (SAL) [34] (in this case, using a Java program) using the EOFMC formal semantics in virtually the same manner as with EOFMs [26]. The major difference between the EOFMC and EOFM translations is how communications are handled; that is, how actions in a *com* decomposition (not present in the EOFM notation) transition out of the ready state. In the EOFMC translation, when the *StartCondition* of a human communication action is satisfied, all variables representing actions in the associated *com* decomposition are set to *Done*. At the same time, the variable representing the communication value is set to the value being communicated, and the local variables human operators use to receive the communication are set to the communicated value.

Figure 2 shows the pattern of SAL notation used to represent a normative communication. In this, `[] StartCondition -->` represents the nondeterministic guard that must be satisfied for the transition to occur. An apostrophe appended to a variable indicates that the variable's value in the next state is being assigned and/or referenced. A *com* decomposition will contain a human communication action `HumanComAction` and `N` local variable actions `LocalVariableAction1-LocalVariableActionN`, where `N` is a positive integer. Each action has an associated variable containing a value. `ComActionValue` represents the value being communicated by the human communication action and `LocalVariableValue1-LocalVariableValueN` represent the values associated with local variable actions `LocalVariableAction1-LocalVariableActionN`, respectively.

The translated EOFMC can be integrated into a larger system model using a defined architecture and coordination protocol [26,35]. Formal verifications are performed using SAL's Symbolic Model Checker (SAL-SMC). Any produced counterexamples can be visualized and evaluated using the EOFMC's visual notation (extended from [36]). In a visualized counterexample, each step is drawn on a separate page of a document. Any task with executing activities or actions is drawn next to a listing of other model variables and their values at the given step. Any drawn task hierarchy will have the color of its activities and actions coded to represent their execution state at that step. Other listed model variables can be categorized based on the model concept they represent: human mission, human-automation interface, automation, environment, and other. Any changes in act execution state or variable values from previous steps are highlighted. More information on the visualizer can be found in [36].

C. Miscommunication Generation

There are many reasons why human–human miscommunication can occur (see [37]). From an engineering and design perspective, a miscommunication can manifest in one of three ways [38]: it can be an "action failure," where the communicator does not communicate the correct information; a "misperception," where the recipient of the communication does not correctly receive the communicated information; or both. To support miscommunication generation, the translator was modified. When miscommunication generation was enabled, an additional transition

[†]An instantiated EOFMC is a task model implemented using the EOFMC.

```
[]StartCondition -->
    HumanComAction'   = Done;
    ComActionVal'     = ComVal;
    LocalVarAction1'  = Done;
    LocalVar1'        = ComActionVal';
    ...
    LocalVarActionN'  = Done;
    LocalVarN'        = ComActionVal';
```

Fig. 2 SAL code [34] pattern generated for each normative human–human communication from an EOFMC.

```
[]StartCondition AND (Count < ComErrorMax) -->
    HumanComAction'   =  Done;
    ComActionVal'     IN {x: ComType | TRUE};
    LocalVarAction1'  =  Done;
    LocalVar1'        IN {x: ComType | TRUE};
    ...
    LocalVarActionN'  =  Done;
    LocalVarN'        IN {x: ComType | TRUE};
    Count'            =  IF ComActionVal' /= ComVal
                            OR LocalVar1' /= ComVal
                            OR ...
                            OR LocalVarN' /= ComVal
                         THEN Count + 1
                         ELSE Count ENDIF;
```

Fig. 3 SAL code pattern generated for each human–human miscommunication from an EOFMC.

(one beyond the transition based on the pattern in Fig. 2) for each original communication transition was generated using the pattern shown in Fig. 3.

In these transitions, in every case where the communicated value would have normatively been assigned to a variable, the variable can assume any value that can be communicated through the associated communication action (Fig. 3). To generate miscommunication, every variable from the original, nominal transition that would have been assigned `ComVal` or `ComActionVal'` (any variable representing the transfer of communicated information) is now assigned a value using the expression `IN {x: ComType | TRUE}`. This means that any variable representing the communicated information can assume any possible value of `ComType`. Therefore, the communicated information (what is assigned to `ComActionVal'`) and the perceived communicated information (what is assigned to `LocalVar1'-LocalVarN'`) can all potentially be erroneous. Further, `Count` represents the total number of miscommunications that have occurred and `ComErrorMax` represents the total number of miscommunications that are allowed to occur. The presence of the assertion `Count < ComErrorMax` in the transition's guard ensures that the transition will never occur if `Count` has reached `ComErrorMax`. The `IF ... THEN ... ELSE ... ENDIF` statement used to assign a value to `Count` only allows it to be incremented if a miscommunication has actually occurred (if the variables representing communicated information are not equal to `ComVal` in the next state). Note that $/=$ represents a "not equals" Boolean operator.

Thus, not only can the communicator improperly communicate the information, but each of the recipients can improperly receive it. Additionally, to give analysts control over the total number of miscommunications, the method has a constant maximum (`ComErrorMax`) and a counter (`Count`) to track the number of miscommunications. The guard condition ensures that a miscommunication transition can only occur if the maximum has not been reached.

V. Application

To illustrate how the approach can model a safety-critical human–human communication protocol, a series of instantiated EOFMCs were constructed for an aircraft heading change in an air traffic control example [24]. The associated communication protocols were evaluated with the method to iteratively determine whether or not they were robust for increasing numbers of miscommunications. This application is meant to be generic (not specific to any given aircraft), and thus may not reflect protocol behavior in any particular aircraft. As such, the presented protocols would need to be adapted to the individual differences associated with different aircraft and airline procedures. This is explored more deeply in Sec. VII.A.

This application was first introduced in [24] and further developed in [27]. Because the application was verified to be true under normative conditions [24] (without miscommunication), it is an interesting application here, as it will demonstrate the added utility provided by the miscommunication generation.

This application has three human operators: a pilot flying (PF), a pilot monitoring (PM), and an air traffic controller (ATCO), where the air traffic controller wants to clear the aircraft being flown by the pilots to a new heading. All three humans have switches they can press when they want to verbally communicate over the radio. They release the switch when they have finished talking.

This example also assumes that the pilots have a mode control panel with a heading window (Fig. 4). In our assumed example and flight mode conditions, a digital display indicates the selected heading in degrees (between 0 and 359). The desired heading can be changed by rotating a heading select knob and pulling it out. Once this is completed, the heading can be engaged so that the selected heading is communicated to the autopilot.

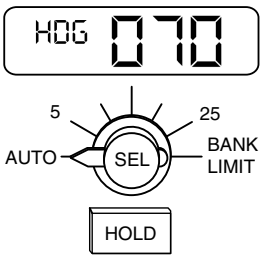

Fig. 4 Heading window on the aircraft's mode control panel.

The following discusses a sequence of human–human communication protocols that were designed to ensure that a correct heading was always communicated and set. In all cases, the method is used to discover the maximum number of miscommunications that cause the protocol to fail. Each iteratively presented protocol is designed to be more robust than the previous one.

A. Protocol 1

The following describes the first communication protocol designed to ensure that this heading is correctly communicated from the air traffic controller to the two pilots [24]:

1) The air traffic controller contacts the pilots and communicates a new heading to them.
2) The PM, in turn, contacts the ATCO and reads the heading back to her.
3) If the heading the ATCO hears back from the pilots is not the one she intended, then this process is repeated (starting at step 1) until the correct heading is communicated back.
4) The PF then dials the new heading.
5) Before the heading is engaged, the PM points at the heading window and reads off the entered heading.
6) If the heading the PF hears read back by the PM does not match the heading that the PF heard from the ATCO, the process for entering and confirming the heading must be repeated (starting with step 4).
7) The PF engages the selected heading.

The following demonstrates how this protocol can be instantiated in an EOFMC and how formal verification can be used to prove whether it will always ensure that the correct heading is engaged.

1. *Modeling*

This communication protocol was instantiated as an EOFMC (Fig. 5; [24]) with the three human operators (the ATCO, the PF, and the PM) all sharing a single task.[‡] The protocol starts when the ATCO presses her push to talk button. The controller then communicates the desired heading (*lATCOSelectedClearance*) verbally to the pilots (the *hATCOTalk* human communication action). Both pilots hear and remember this heading: this is modeled as the heading being stored in the local variables *lPFHeadingFromATCO* and *lPMHeadingFromATCO* for the PF and PM, respectively. The ATCO releases the switch. Next, the PM presses his switch and repeats/communicates the heading that was heard, where both the ATCO and PF hear and remember the heading that was read back. The PM releases the switch. This entire process must repeat if the heading the ATCO hears from the PM does not match the heading she wanted to communicate ($lATCSelectedClearance \neq lATCHeadingHeardFromPilots$). Otherwise, it completes.

To keep the size of the models manageable, the headings in all of the modeled protocols presented in this paper are represented abstractly as an enumerated type. A heading is correct (*CorrectHeading*) if the heading matches the clearance intended to be communicated by the ATCO. It is incorrect if it does not match the heading clearance the ATCO intended to communicate. Three abstract values for an incorrect heading were used to allow a miscommunication to occur. with all three participants having the miscommunication manifest differently.

Once the heading has been read back correctly, the pilots collaborate to set the new heading (*aSetNewHeading*). This requires the new heading to be selected and confirmed (*aChangeAndConfirm*) and then executed (*aExecuteTheChange*). The activity *aChangeAndConfirm* starts with the PF pushing and rotating the heading select knob to the heading heard from the ATCO and then pulling the knob. The PM verifies that the PF has dialed the correct heading and confirms the heading selection by pointing to the heading in the mode control panel (MCP) heading window and verbally communicating the entered value. Here, two communications can occur separately or in parallel (indicated by *aConfirmTheChange*'s *and_par* decomposition operator): the PM points at the heading window (*aPointAtHeadingWindow*) and speaks the heading that was entered (*aSayTheHeading*). Both are perceived by the PF. This process must repeat if the heading the PF hears spoken by the PM does not match the heading that was heard from the ATCO ($lPFHeadingFromPM \neq lPFHeadingFromATCO$). Once the heading is confirmed, the PF presses the heading select button to execute the heading change.

2. *Translation*

The instantiated EOFMC was translated into SAL using the automated translator. The original model contained 144 lines of extensible markup language (XML) code. The translated model contained 404 lines of SAL code. This model was asynchronously composed with the model representing the MCP heading window, where the cockpit's displayed heading could be changed when the pilot rotated the heading knob. These two models were asynchronously composed together to create the full system model. The system model was then used to create two different versions: one where the maximum number of miscommunications (`ComErrorMax`) was set to zero, and one where it was set to one.

3. *Specification and Verification*

The purpose of the communication protocol is to ensure that the pilots set the aircraft heading to that intended by the air traffic controller. Thus, linear temporal logic [39] (the specification logic used by SAL) can express this as

$$\boldsymbol{G}\left(\begin{array}{l}(aChangeHeading = Done)\\ \Rightarrow (iHeadingWindowHeading = lATCSelectedClearance)\end{array}\right) \tag{1}$$

This asserts that, for all paths through the model, it will always be true ($\boldsymbol{G}$) that, when the activity for changing the heading is done, the heading in the MCP heading window will match the clearance selected by the ATCO.

Further, to ensure that Eq. (1) was not true due to vacuity, the following specification was also used:

$$\boldsymbol{F}(aChangeHeading = Done) \tag{2}$$

This asserts that, for all paths through the model, it will eventually be true ($\boldsymbol{F}$) that the activity for changing the heading will be done.

Finally, deadlock checking [40] was performed to ensure that no deadlock states were present in the protocol.

These specifications were checked against the two versions of the formal model using SAL-SMC on a workstation with 64 GB of RAM, dual 3.3 GHz quad-core Intel Xeon processors, and the Linux Mint 14 desktop.

[‡]In all EOFM visualizations (Figs. 5–7), activities are prefixed by "*a*", actions by "*h*", inputs by "*i*", and local variables by "*l*". Values or variables used in a communication action are bolded.

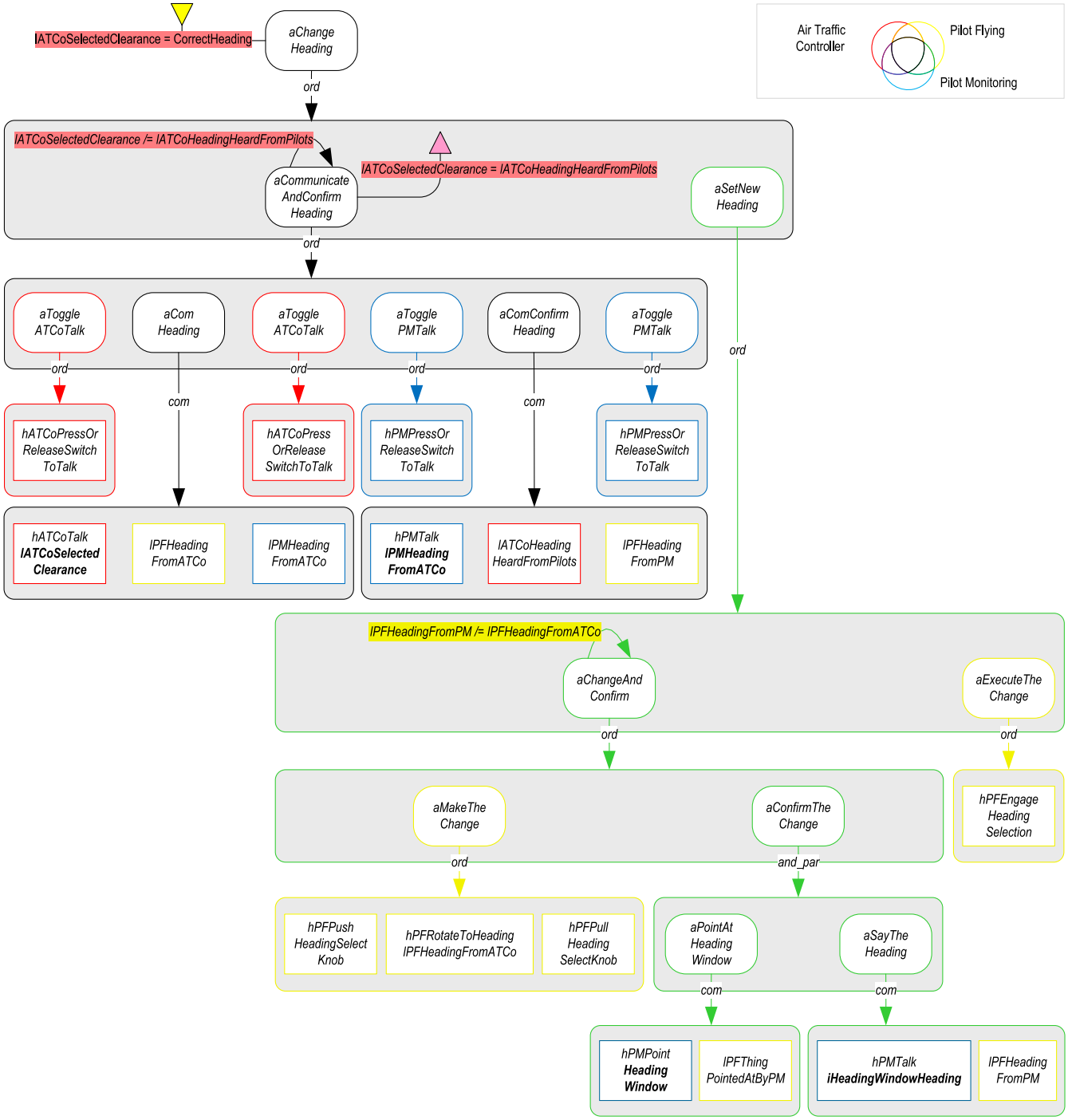

Fig. 5 Visualization of the EOFMC representing Protocol 1 [24].

The first model (`ComErrorMax = 0`), verified to true in 1.74 s (total execution time), having visited 6410 states (this is consistent with the results presented by [27]). The second model (`ComErrorMax = 1`) returned a counterexample after 2.07 s (total execution time), having visited 75,450 states.[§]

4. *Failure Diagnosis*

To help diagnose why this failure occurred, the counterexample was visualized using the technique described in [36]. This revealed the following failure sequence:

1) The air traffic controller, wanting to clear the aircraft to a new heading (*CorrectHeading*), presses the switch to talk.
2) When the air traffic controller issues the clearance, a miscommunication occurs: the PF hears *IncorrectHeading3* and the PM hears *CorrectHeading*.
3) The air traffic controller releases the switch.
4) The PM presses his switch to talk.
5) The PM repeats back the correct heading heard from the air traffic controller without a miscommunication occurring.

[§]The specification in Eq. (2) verified to true for both models and no deadlock states were detected for either model.

6) The PM releases the switch to talk.

7) Since the correct heading was heard by the PM, the air traffic controller allows the activity for communicating the heading (*aCommunicateAndConfirmHeading*) to complete its execution.

8) The pilots begin collaborating to enter the heading from the air traffic controller.

9) The PF performs the activity for changing the heading: pressing the heading select knob, setting the dial to the heading heard from the air traffic controller (*IncorrectHeading3*), and pulling the heading select knob.

10) The PM then points at the heading display and reads off the heading entered (*IncorrectHeading3*) to the PF.

11) Because the incorrect heading just heard from the PM (*lPFHeadingFromPM2*) matches the incorrect heading the PF heard from air traffic control (*lPFHeadingFromATC*), the PF engages the new (incorrect) heading.

Thus, although specifically designed to protect against miscommunication, protocol 1 does in fact allow an incorrect heading to be engaged.

B. Protocol 2

The failure discovered in protocol 1 appears to occur because it did not give the two pilots a means of reconciling differences between the headings they heard from air traffic control. Thus, protocol 2 was developed to hopefully correct this and allow for such a reconciliation to occur by allowing the pilots to request that the ATCO repeat the heading. Thus, protocol 2 proceeds as follows:

1) The air traffic controller contacts the pilots and gives them a new heading clearance.

2) The PF goes through the process of entering the new heading.

3) Then, the pilots collectively check the heading to ensure that what was entered matches what they heard from the ATCO.

4) If the entered heading matches the heading the PM heard from the ATCO, the PM goes through the procedure of reading back the entered heading to the ATCO. This readback process must repeat until the heading the PF hears read back to the ATCO by the PM matches what was entered in the heading window. This is meant to ensure that the PM is reading back the heading both heard from ATC and entered in the MCP heading window. After this process, if the heading the ATCO heard read back by the pilots does not match the originally selected heading, the process must restart at step 1.

5) Alternatively, if the entered heading does not match the heading the PM heard from the ATCO, the PM will request that the ATCO repeat the heading. If this occurs, the protocol restarts at step 1.

6) If the preceding steps do not produce a request for the ATCO to repeat the heading and the heading the ATCO hears read back matches the original selected heading, then the PF can engage the entered heading.

1. *Modeling and Translation*

Protocol 2 was instantiated as an EOFMC (Fig. 6). This was then automatically translated into the input language of SAL. Where the original model contained 149 lines of XML markup, the translated version was 393 lines of SAL code. The translated version of the EOFMC instance was synchronously composed with the heading window behavior model in the same way as protocol 1.

2. *Verification*

The SAL representation of protocol 2 was checked against Eq. (1) using SAL-SMC for three different versions of the model: one with no miscommunications (`ComErrorMax = 0`), one with a maximum of one miscommunication (`ComErrorMax = 1`), and one with a maximum of two miscommunications (`ComErrorMax = 2`). In these analyses, the first two models verified to true in 1.72 and 1.8 s (total execution time), having visited 6282 states and 60,207 states, respectively. However, the model with `ComErrorMax = 2` failed its verification after visiting 131,614 states in 1.85 s and returned a counterexample.[¶]

3. *Failure Diagnosis*

The counterexample was evaluated with the counterexample visualizer [36]. This illustrated a failure sequence that proceeded as follows:

1) The air traffic controller, wanting to clear the aircraft to a new heading (*CorrectHeading*), presses the switch to talk.

2) When the air traffic controller issues the clearance, a miscommunication occurs and both pilots hear *IncorrectHeading3*.

3) The air traffic controller releases the switch.

4) The PF performs the activity for changing the heading: pressing the heading select knob, setting the dial to the heading heard from the air traffic controller (*IncorrectHeading3*), and pulling the heading select knob.

5) The pilots then start performing the activity for checking the entered heading (*aCheckHeading*). Because the incorrect heading in the heading window matches the heading the PM incorrectly heard from the ATCO, the pilots perform a readback of the entered heading (*aReadBack*).

6) The PM presses his switch to talk.

7) The PM repeats back the incorrect heading heard from the air traffic controller. However, a miscommunication occurs and the ATCO hears the correct heading instead of the communicated incorrect heading.

8) The PM releases the switch to talk.

9) Since the correct heading was heard by the ATCO, the air traffic controller allows the activity for communicating the heading (*aCommunicateAndConfirmHeading*) to complete its execution.

10) The PF engages the new (incorrect) heading.

The addition of the pilot's ability to reconcile differences between the headings they heard from the ATCO allowed for an increase in the total number of miscommunications that could be accounted for by the protocol. However, two readbacks were ultimately not enough to ensure that the correct heading was entered.

C. Protocol 3

The failure sequence found for protocol 2 occurs because the two miscommunications allow the same incorrect heading to be heard by the pilots on the initial communication and the correct heading to be incorrectly heard by the ATCO on the readback. Thus, in the third protocol, an attempt to develop a more robust procedure is presented that uses three readbacks. Since such a protocol requires a significant amount of radio communication, the procedure for pilots reconciling differences in the headings each heard from the ATCO has been modified so that it does not require radio communication (as it did in protocol 2).

[¶]The specification in Eq. (2) verified to true for all three models and no deadlock states were found.

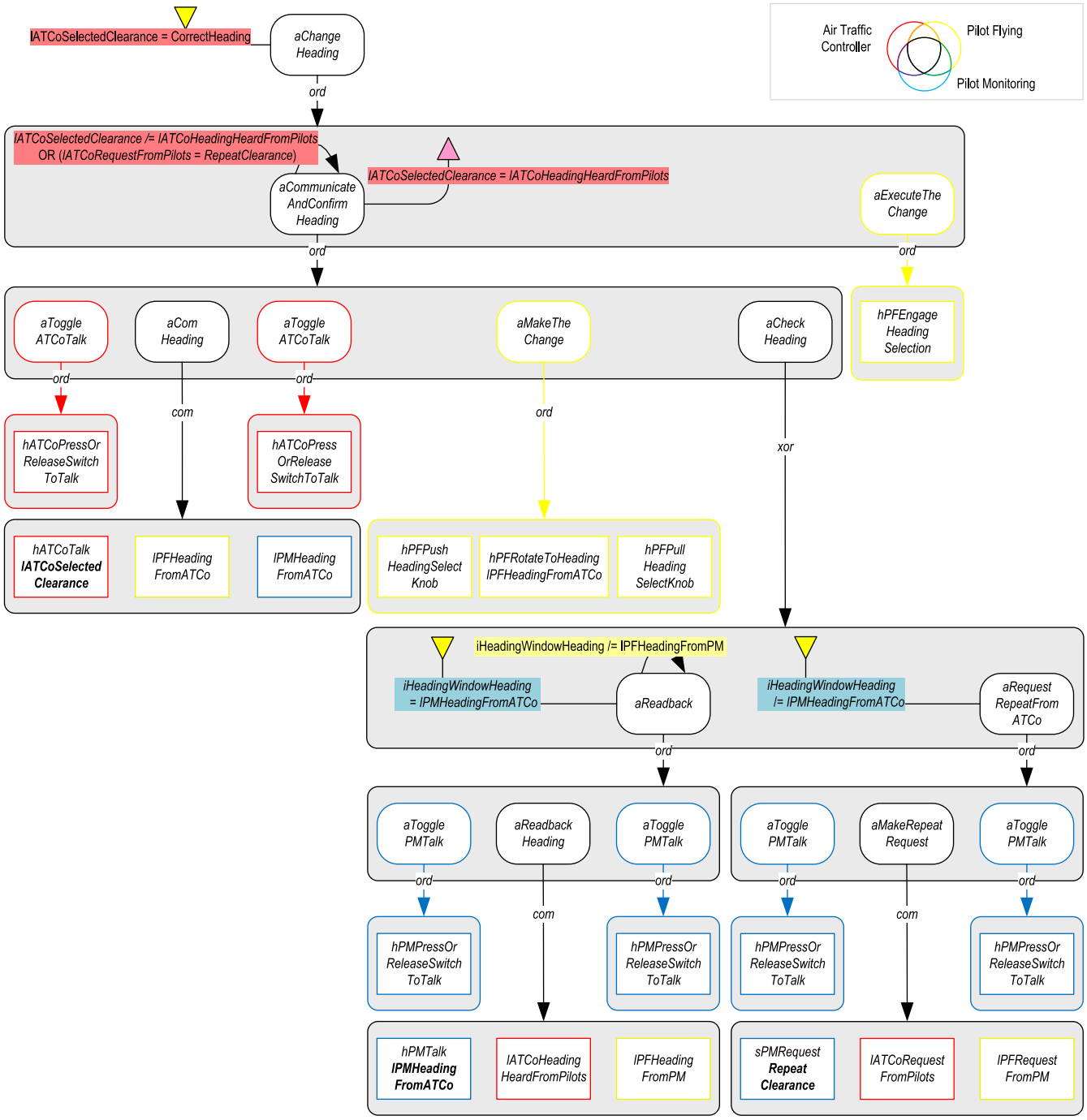

Fig. 6 Visualization of the EOFMC representing Protocol 2.

Protocol 3 proceeds as follows:

1) The air traffic controller contacts the pilots and gives them a new heading clearance.

2) The PF goes through the process of entering the new heading.

3) The pilots collectively check the heading to ensure that what was entered matches what they heard from the ATCO.

4) If the entered heading does not match the heading the PM heard from the ATCO, the pilots must decide between themselves which of the two headings to use. If they decide to use the heading already entered, then the protocol proceeds at step 5. Otherwise, the PF must enter the new heading (the one the PM originally heard from the ATCO) at step 2. Note that this is unlikely behavior in current air traffic protocols. If the pilots find themselves in disagreement, they will request a readback from ATC as they did in protocol 2. However, protocol 3 is meant to represent a new type of procedure where this step would not be necessary, as the correctness of the entered heading should be determined from subsequent readbacks.

5) The pilots go through the procedure for reading back the entered heading to the ATCO. This readback process must repeat until the heading the PF heard read back to the ATCO by the PM matches what was entered in the heading window.

6) The ATCO then performs the process of reading back the selected heading clearance to the pilots.

7) If this last heading read back to the pilots does not match the heading entered in the MCP heading window, the whole process must begin again at step 3.

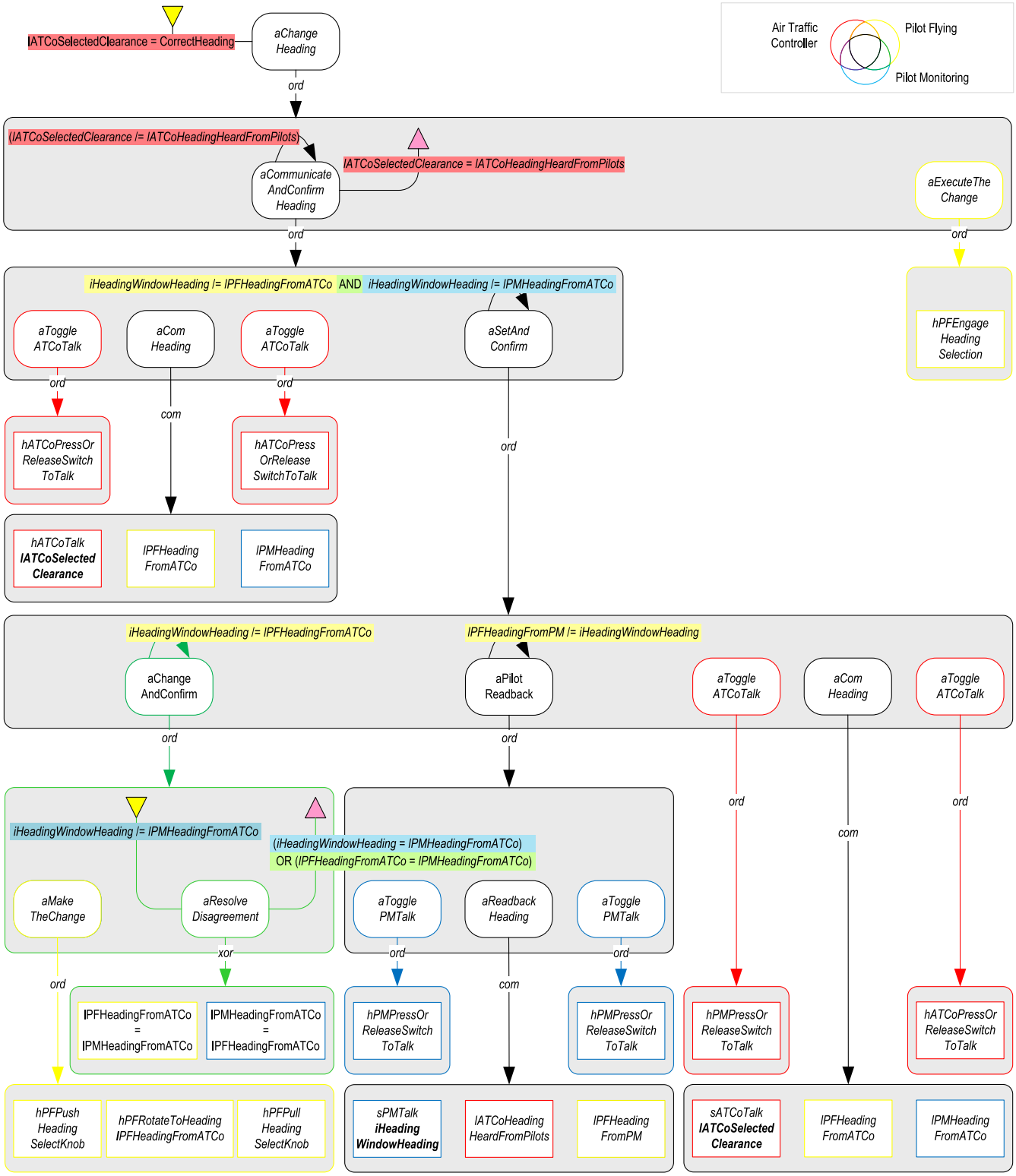

Fig.7 Visualization of the EOFMC representing Protocol 3.

8) Alternatively, if the entered heading does not match the heading the PM heard from the ATCO, the PM will request that the ATCO repeat the heading. If this occurs, the protocol restarts at step 1.

9) The PF engages the entered heading.

1. *Modeling and Translation*

Protocol 3 was instantiated as an EOFMC (Fig. 7). Note that, in this model, the pilot collaboration is modeled in the *aResolveDisagreement* activity. In this, the product of the disagreement resolution is modeled by reassigning either the heading the PF heard from the ATCO or the heading the PM heard from the ATCO to match the heading heard by the other.

The EOFMC representation of the protocol was then translated into SAL's input language. Where the original EOFMC instance contained 149 lines of XML markup, the translated version had 449 lines of SAL code. The translated version of the model was synchronously composed with the heading window behavior model in the same way as protocol 1 and protocol 2.

2. Verification

Protocol 3 was checked against Eq. (1) using SAL-SMC for three different versions of the model: one with no miscommunications (`ComErrorMax = 0`), one with a maximum of one miscommunication (`ComErrorMax = 1`), one with a maximum of two miscommunications (`ComErrorMax = 2`), and one with a maximum of three miscommunications (`ComErrorMax = 3`). In these analyses, the first three models verified to true in 2.01, 2.52, and 2.73 s (total execution time), having visited 25,893, 35,043, and 144,305 states, respectively. However, the model with `ComErrorMax = 3` failed its verification after visiting 262,423 states in 2.85 s and returned a counterexample.

3. Failure Diagnosis

The counterexample was processed through the counterexample visualizer [36]. This helped us derive the following sequence of events that produced the failure:

1) The air traffic controller, wanting to clear the aircraft to a new heading (*CorrectHeading*), presses the switch to talk.
2) When the ATCO issues the clearance, a miscommunication occurs and both pilots hear *IncorrectHeading3*.
3) The ATCO releases the switch.
4) The PF performs the activity for changing the heading: pressing the heading select knob, setting the dial to the heading heard from the ATCO (*IncorrectHeading3*), and pulling the heading select knob.
5) Because the heading entered into the heading window matches what the PM heard from the ATCO, there was no need for the pilots to resolve disagreement (*aResolveDisagreement*). Thus, the pilots proceed to perform the heading readback.
6) The PM repeats back the incorrect heading heard from the ATCO. However, a miscommunication occurs and the ATCO incorrectly hears the correct heading (*CorrectHeading*).
7) The PM releases the switch to talk.
8) Having heard the correct heading read back to him, the ATCO then initiates the final readback to the pilots by pressing the switch to talk.
9) When the ATCO repeats the correct clearance, a miscommunication occurs and both pilots once again hear *IncorrectHeading3*.
10) The ATCO releases the switch.
11) Since both pilots heard the heading that was entered read back to them, the PF engages the entered (incorrect) heading.

VI. Verification Benchmarks

To get a sense of how the approach scales for realistic applications, benchmarks were collected for all three of the communication protocols with between zero and three maximum miscommunications. Because Eq. (1) did not verify to true for all of the trials considered in this experiment, statistics (number of visited states and total verification time) were collected for the verification of Eq. (2) that did. These are reported in Fig. 8.

When these results were fitted to an exponential function ($\hat{y}$ equations in Fig. 8), all fitted with $R^2 \geq 0.89$. All but one had $R^2 \geq 0.96$. Thus, both the size of the model's state space and the verification time appear to increase exponentially with the maximum number of miscommunications.

VII. Discussion and Future Work

Human–human communication protocols can be critical to the safe operation of a system and can fail in unexpected ways. This paper introduced a method that allows human–human communication protocols to be evaluated with model checking. Because human communications during coordinated activities can include actions for communicating both verbal and nonverbal information and result in noncommunication human actions, this work considers human communication as part of human task behavior. The EOFMC was used to represent communication protocols as shared task behaviors that include synchronous verbal communications, gestures, activities, and low-level actions, as well as asynchronous human behaviors associated with the communication. The EOFM was described along with its formal semantics, and the process was used to translate the formal models into a model-checking language. A new method for automatically generating miscommunications

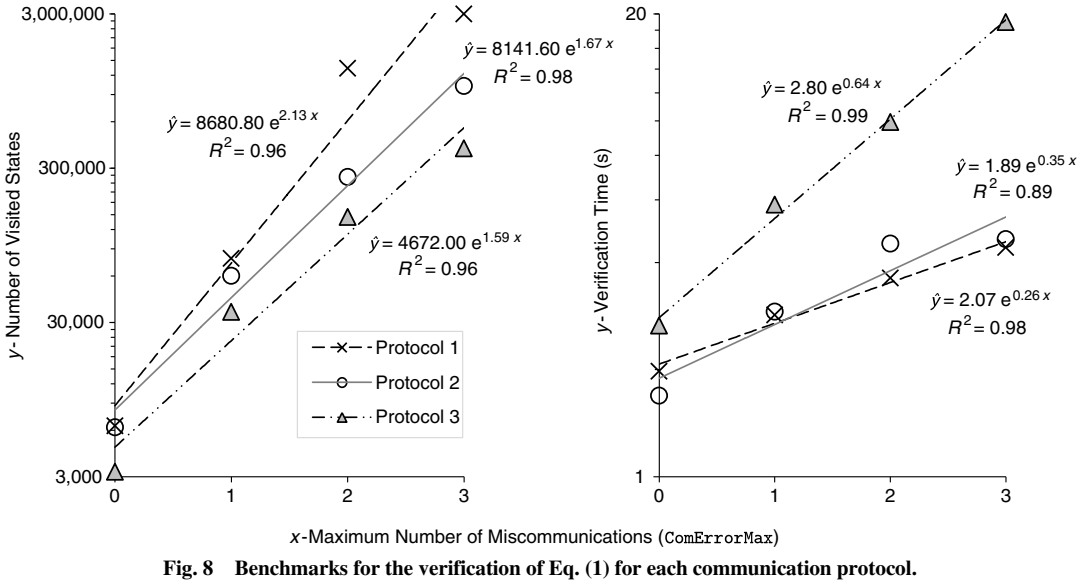

Fig. 8 Benchmarks for the verification of Eq. (1) for each communication protocol.

between human operators was introduced, and it was implemented in a translator, enabling the miscommunications to be automatically included in the translated representation of an instantiated EOFMC protocol. Because of the theoretical [38] and generic nature of the miscommunication-generation approach, the presented method gives analysts the means to evaluate how robust to miscommunications different protocols are. This was demonstrated by applying the method to the evaluation of different protocols in a safety-critical air traffic control application. Benchmarks were reported that showed how the method scales with both the size of the EOFMC and the maximum number of miscommunications.

Although not the goal of this work, the presented protocols illustrate how different elements of human–human interaction can be modeled in EOFMC. Protocol 1 (Fig. 5) shows how both verbal and nonverbal (pointing) communications can be represented. Protocol 2 shows how multiple types of communication (heading communication and readback requests) can be modeled and how two different people can initiate the execution of a shared activity (*aReadback* from Fig. 6, where both the PM and PF can initiate the readback based on the value in the heading windows and what was heard read back to the ATCO, respectively). Finally, protocol 3 (Fig. 7) demonstrates how discrepancies could be resolved between humans. Specifically, *aResolveDisagreement* shows how the EOFMC can be used to model pilots attempting to reconcile differences between their perceptions. In this case, this is represented abstractly as the pilots reaching one of two possible conclusions to such an activity (setting the appropriate local variable to be consistent with what the other pilot heard). Thus, even with the simple means it uses to represent communication, the EOFMC is capable of representing a wide range of human communication and coordination behavior.

Although the method has shown itself to be successful here, there are still a number of places for improvement and future development. These are discussed in the following subsections.

A. Tradeoffs Between Protocols and Additional Applications

It is interesting to note that, in the presented application, as the protocols became more robust to miscommunications, the number of readbacks required between the pilots and ATC increased from one in protocol 1 to a minimum of two in protocol 3 (one from the pilots and one from the ATCO). It is conceivable that additional readbacks would improve the robustness of the protocol to higher maximum numbers of miscommunications. Obviously, air traffic controllers and pilots are busy with their other responsibilities, and thus will want to keep the complexity and time required for performing the protocol to a minimum. Thus, there is a tradeoff between these factors and the robustness of the protocol. Future work should attempt to identify the optimal number of readbacks to ensure the desired compromise between safety and performance.

Different types of air traffic control miscommunication are more likely than others [5,41]. Thus, in situations where analysts cannot add additional readbacks to improve robustness, they may wish to only consider probable miscommunications. Future work should investigate how this functionality could be adapted to the presented method.

Additionally, the protocols explored in this paper were intentionally generic: not relating to any particular aircraft of airline policy. To be genuinely useful, they would need to be adapted to specific airline procedures and aircraft. Thus, although the presented protocols could provide guidance for protocol designers, any protocols derived from those presented here should be evaluated with the method to ensure they maintain their robustness.

Further, the example presented here represents only a small subset of the collaborative procedures used by pilots and ATC in the aerospace system. There are also many other domains where human–human communication and collaboration protocols are critical to safe operations. These domains may have different procedures for ensuring accurate information is conveyed. Future work should explore how the EOFMC can be used to check protocols in other domains.

B. Scalability

The exponential increases observed in state-space size and verification time for increases in `ComErrorMax` (Fig. 8) suggest that there could be scalability problems associated with larger communication protocols with large `ComErrorMax` values, EOFMC instances that contain many more shared or unshared task behaviors, and/or complex systems that the EOFMC instance would interact with. Thus, scalability could limit what types of human–human communication protocols the method could be used to evaluate and/or the maximum number of miscommunications that could be considered in a given verification. Given that the EOFMC to SAL translator explicitly represents the execution state of each activity and action, and transitions between each of these independently and in the context of the task hierarchy, it is likely that the formal representation is state-space inefficient. Future work should investigate how EOFMC scalability can be improved by compacting these representations.

Alternatively, other types of automated theorem provers can help analysts perform formal verification, and such tools have been used with human-interactive systems (see [22] for a review). It is true that these tools can often avoid the scalability limitations of model checking. However, they can also be much more difficult to use. Future work should investigate the scalability/usability tradeoffs of using automated theorem provers to formally verify human–human communication protocols.

C. Other Modeling Formalisms

SAL was used in this work because the EOFM, the base task analytic modeling formalism, uses SAL. This made it easy to adapt the existing translation tools for use in the EOFMC. However, it is possible that other formal tools could prove to be better for this particular application. For example, Communicating Sequential Processes (CSP) [42] natively models communication protocols, and thus may be better suited to this work. Alternate formal modeling languages and tools could conceivably help address the scalability concerns discussed previously. Alternate formal modeling infrastructures should be considered in future work.

Alternatively, other types of automated theorem provers can help analysts perform formal verification, and such tools have been used with human-interactive systems (see [22] for a review). It is true that these tools can often avoid the scalability limitations of model checking. However, they can also be much more difficult to use. Future work should investigate the scalability/usability tradeoffs of using automated theorem provers to formally verify human-human communication protocols.

D. Miscommunication and Communication Failure Extensions

When generating miscommunications using the presented method, all of the ways that a miscommunication could manifest are considered to be equally probable. However, in reality, certain miscommunications will be more likely than others [37]. For example, it is much more likely that a heading clearance will be misheard as a similarly sounding heading as opposed to one that sounds nothing like the actual heading. Similarly, the target of a human's pointing communication would be more likely to be misinterpreted as something in the target's periphery rather than something further away. Thus, there could potentially be a number of miscommunications the method considers that analysts may not find probable enough to be worth including. Eliminating unlikely miscommunications could help improve the scalability of the method while improving its utility. Future work should attempt to extend the method to include this feature.

There are also different ways that communication failures can occur. For example, Sperber and Wilson [43] developed the "code model" of communication. This posits that human–human communication occurs when the communicator encodes a message and sends it to a recipient over or through a channel. This suggests several different ways communication failures can manifest [44]: 1) the wrong message is encoded; 2) the correct message is wrongly encoded; 3) the channel improperly transmits the encoded message; 4) the message is decoded incorrectly by the recipient.; and/or 5) the recipient misunderstands the message.

Assuming that a message is communicated, the method should be able to approximate the result of all of these failures (that an incorrect message was communicated). However, there may be utility in modeling communications at a lower level than currently associated with this taxonomy. This should be the subject of future research. Further, implicit within item 3 of the "code model" is the notion that a channel could produce both an incorrect communication as well as no communication. Although the current miscommunication-generation system can replicate the former, it cannot reproduce the latter. Future work should investigate how to include failed communications (conditions where no information is conveyed) in the generation method.

E. Other Erroneous Human Behavior Considerations

Miscommunication is only one type of erroneous human behavior that could impact the success of the task associated with a human–human communication protocol. For example, even when a human operator is aware of how to properly perform a task, failures of memory, attention, or human coordination can cause him or her to perform actions or activities incorrectly [45,46]. Related work has investigated how to generate erroneous human behavior in task analytic models and evaluate its impact on systems using model checking [32,47]. Thus, it should be possible to evaluate how robust human–human communication protocols are to these other types of erroneous human behavior, both with and without miscommunication. Future work should investigate how to include both types of behavior generation in analyses synergistically.

Further, the types of erroneous behaviors that have been included in task analytic models and evaluated formally have almost exclusively focused on the behavior of a single human operator [32,33,47–50]. Human–human communication protocols can have each of the human operators doing specific elements of a task on his or her own but also require coordinated behavior between the different human participants. To date, no work has focused on how to model problems with human–human coordination. Future work should investigate this subject.

F. Comparison to Simulation Models

A number of environments and cognitive architectures exist that allow human behavior and human–human communication to be evaluated using simulation [51–54]. Future work should compare the method with these, determine what the tradeoffs are between them, and investigate possible avenues of synergy.

VIII. Conclusions

This work has presented a novel method for using model checking and task analytic behavior modeling to evaluate how robust human–human communication protocols are to miscommunications. Further, we determined the scalability limitations of this method and presented it usefulness through its iterative application to human–human communication protocols used to convey heading clearances from air traffic control to pilots. This work is significant because it gives analysts an unprecedented ability to evaluate the safety of human communication protocols, while considering human–human communication and coordination as part of its larger human task.

Acknowledgments

The project described was supported by NASA award NNA10DE79C. The content is solely the responsibility of the author and does not necessarily represent the official views of NASA. The author would like to thank Ellen Bass for her advice in the preparation of this manuscript.

References

[1] "Pilot-Controller Communication," *Flight Safety Digest*, ALAR Briefing Note, Aug.–Nov. 2000, pp. 47–53.

[2] "Effective Pilot/Controller Communications," *Human Performance*, Flight Operations Briefing Notes, Airbus Industries, Blagnac Cedex, France, 2006, pp. 1–17.

[3] *Aeronautical Information Manual: Official Guide to Basic Flight Information and ATC Procedures*, Federal Aviation Administration, U.S. Dept. of Transportation, 2014, http://www.faa.gov/air_traffic/publications/media/AIM_Basic_4-03-14.pdf [retrieved 2014].

[4] Jones, R. K., "Miscommunication Between Pilots and Air Traffic Control," *Language Problems and Language Planning*, Vol. 27, No. 3, 2003, pp. 233–248.

[5] Barshi, I., and Farris, C., *Misunderstandings in ATC Communication: Language, Cognition, and Experimental Methodology*, Ashgate Publishing, Surrey, England, U.K., 2013, p. 270.

[6] Buerki-Cohen, J., "An Analysis of Tower (Ground) Controller-Pilot Voice Communications," Federal Aviation Administration, U.S. Dept. of Transportation TR-DOT/FA/4AR-9619, DOT-VNTSC-FAA-95-41, 1995.

[7] "Pilot/Controller Communications," NASA Aviation Safety Reporting System NASA Ames Research Center, TR-TH:262-7, Moffett Field, CA, 2014, http://asrs.arc.nasa.gov/docs/rpsts/plt_ctlr.pdf [retrieved 2014].

[8] Billings, C. E., and Reynard, W. D., "Dimensions of the Information Transfer Problem," *Information Transfer Problems in the Aviation System*, NASA Ames Research Center, Moffett Field, CA, 1981, pp. 9–14.

[9] Wing, J. M., "A Specifier's Introduction to Formal Methods," *Computer*, Vol. 23, No. 9, 1990, pp. 8, 10–22, 24.

[10] Clarke, E. M., Grumberg, O., and Peled, D. A., *Model Checking*, MIT Press, Cambridge, MA, 1999, p. 314.

[11] Edelkamp, S., Leue, S., and Lluch-Lafuente, A., "Directed Explicit-State Model Checking in the Validation of Communication Protocols," *International Journal on Software Tools for Technology Transfer*, Vol. 5, No. 2, 2004, pp. 247–267.

[12] Argón, P., Delzanno, G., Mukhopadhyay, S., and Podelski, A., "Model Checking Communication Protocols," *Proceedings of the 28th Conference on Current Trends in Theory and Practice of Informatics*, Springer, Berlin, 2001, pp. 160–170.

[13] Sunshine, C. A., "Formal Methods for Communication Protocol Specification and Verification," RAND Corp. TR-N-1492-ARPA/NBS, Santa Monica, CA, 1979.

[14] Bochmann, G., and Sunshine, C., "Formal Methods in Communication Protocol Design," *IEEE Transactions on Communications*, Vol. 28, No. 4, 1980, pp. 624–631.

[15] Pek, E., and Bogunovic, N., "Formal Verification of Communication Protocols in Distributed Systems," *Proceedings of MIPRO 2003*, Computers in Technical Systems and Intelligent Systems, MIPRO, 2003, pp. 44–49.

[16] Sidhu, D. P., and Leung, T., "Formal Methods for Protocol Testing: A Detailed Study," *IEEE Transactions on Software Engineering*, Vol. 15, No. 4, 1989, pp. 413–426.
doi:10.1109/32.16602

[17] Dietrich, F., and Hubaux, J., "Formal Methods for Communication Services," Inst. for Computer Communications and Applications, Swiss Federal Inst. of Technology TR-SSC/1999/023, Zurich, 1999.
[18] Austin, J., *How to Do Things with Words: The William James Lectures Delivered at Harvard University in 1955*, Oxford Univ. Press, Oxford, 1962.
[19] Kirwan, B., and Ainsworth, L. K., *A Guide to Task Analysis*, Taylor and Francis, London, 1992, p. 432.
[20] Hörl, J., and Aichernig, B. K., "Formal Specification of a Voice Communication System Used in Air Traffic Control, an Industrial Application of Light-Weight Formal Methods Using VDM++," *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems*, Springer, New York, 1999, pp. 1868–1868.
[21] Hörl, J., and Aichernig, B. K., "Validating Voice Communication Requirements Using Lightweight Formal Methods," *IEEE Software*, Vol. 17, No. 3, 2000, pp. 21–27.
doi:10.1109/52.896246
[22] Bolton, M. L., Bass, E. J., and Siminiceanu, R. I., "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, 2013, pp. 488–503.
[23] Paternò, F., Santoro, C., and Tahmassebi, S., "Formal Model for Cooperative Tasks: Concepts and an Application for En-Route Air Traffic Control," *Proceedings of the 5th International Conference on the Design, Specification, and Verification of Interactive Systems*, Springer, New York, 1998, pp. 71–86.
[24] Bass, E. J., Bolton, M. L., Feigh, K., Griffith, D., Gunter, E., Mansky, W., and Rushby, J., "Toward a Multi-Method Approach to Formalizing Human-Automation Interaction and Human–Human Communications," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, Piscataway, NJ, 2011, pp. 1817–1824.
[25] Paternò, F., Mancini, C., and Meniconi, S., "ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models," *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*, Chapman and Hall, London, 1997, pp. 362–369.
[26] 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*, Vol. 41, No. 5, 2011, pp. 961–976.
doi:10.1109/TSMCA.2011.2109709
[27] Bolton, M. L., and Bass, E. J., "Evaluating Human–Human Communication Protocols with Miscommunication Generation and Model Checking," *Proceedings of the Fifth NASA Formal Methods Symposium*, NASA Ames Research Center, Moffett Field, CA, 2013, pp. 48–62.
[28] Bolton, M. L., and Bass, E. J., "A Method for the Formal Verification of Human Interactive Systems," *Proceedings of the 53rd Annual Meeting of the Human Factors and Ergonomics Society*, Human Factors and Ergonomic Society, Santa Monica, CA, 2009, pp. 764–768.
[29] Bolton, M. L., and Bass, E. J., "Using Model Checking to Explore Checklist-Guided Pilot Behavior," *International Journal of Aviation Psychology*, Vol. 22, No. 4, 2012, pp. 343–366.
doi:10.1080/10508414.2012.718240
[30] Bolton, M. L., and Bass, E. J., "Enhanced Operator Function Model: A Generic Human Task Behavior Modeling Language," *Proceedings of the IEEE International Conference on Systems Man and Cybernetics*, IEEE, Piscataway, NJ, 2009, pp. 2983–2990.
[31] Bolton, M. L., "Automatic Validation and Failure Diagnosis of Human–Device Interfaces Using Task Analytic Models and Model Checking," *Computational and Mathematical Organization Theory*, Vol. 19, No. 3, 2013, pp. 288–312.
[32] Bolton, M. L., Bass, E. J., and Siminiceanu, R. I., "Using Phenotypical Erroneous Human Behavior Generation to Evaluate Human-Automation Interaction Using Model Checking," *International Journal of Human-Computer Studies*, Vol. 70, No. 11, 2012, pp. 888–906.
doi:10.1016/j.ijhcs.2012.05.010
[33] Bolton, M., and Bass, E., "Generating Erroneous Human Behavior from Strategic Knowledge in Task Models and Evaluating Its Impact on System Safety with Model Checking," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 43, No. 6, 2013, pp. 1314–1327.
doi:10.1109/TSMC.2013.2256129
[34] De Moura, L., Owre, S., and Shankar, N., "The SAL Language Manual," Computer Science Lab., SRI International TR CSL-01-01 2(Rev. 2), Menlo Park, CA, 2003, p. 35.
[35] Bolton, M. L., and Bass, E. J., "Formally Verifying Human–Automation Interaction as Part of a System Model: Limitations and Tradeoffs," *Innovations in Systems and Software Engineering: A NASA Journal*, Vol. 6, No. 3, 2010, pp. 219–231.
doi:10.1007/s11334-010-0129-9
[36] Bolton, M. L., and Bass, E. J., "Using Task Analytic Models to Visualize Model Checker Counterexamples," *Proceedings of the 2010 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, Piscataway, NJ, 2010, pp. 2069–2074.
[37] Gibson, W., Megaw, E., Young, M., and Lowe, E., "A Taxonomy of Human Communication Errors and Application to Railway Track Maintenance," *Cognition, Technology and Work*, Vol. 8, No. 1, 2006, pp. 57–66.
doi:10.1007/s10111-005-0020-x
[38] Traum, D., and Dillenbourg, P., "Miscommunication in Multi-Modal Collaboration," *AAAI Workshop on Detecting, Repairing, and Preventing Human–Machine Miscommunication*, AAAI, Palo Alto, CA, 1996, pp. 37–46.
[39] Emerson, E. A., "Temporal and Modal Logic," *Handbook of Theoretical Computer Science*, edited by van Leeuwen, J., Meyer, A. R., Nivat, M., Paterson, M., and Perrin, D., MIT Press, Cambridge, MA, 1990, pp. 995–1072, Chap. 16.
[40] De Moura, L., "SAL: Tutorial," SRI International, Computer Science Lab., Menlo Park, CA, April 2004, http://sal.csl.sri.com/doc/salenv_tutorial.pdf [retrieved 2014].
[41] Scarborough, A., Bailey, L., and Pounds, J., "Examining ATC Operational Errors Using the Human Factors Analysis and Classification System," Federal Aviation Administration, Office of Aerospace Medicine TR-DOT/FAA/AM-05/25, 2005.
[42] Hoare, C. A. R., "Communicating Sequential Processes," *Communications of the ACM*, Vol. 21, No. 8, 1978, pp. 666–677.
doi:10.1145/359576.359585
[43] Sperber, D., and Wilson, D., *Relevance: Communication and Cognition*, 2nd ed., Wiley-Blackwell, New York, 1995, p. 338.
[44] Jones, P. M., "Human Error and Its Amelioration," *Handbook of Systems Engineering and Management*, Wiley, New York, 1997, pp. 687–702.
[45] Reason, J., *Human Error*, Cambridge Univ. Press, New York, 1990, p. 320.
[46] Hollnagel, E., "The Phenotype of Erroneous Actions," *International Journal of Man-Machine Studies*, Vol. 39, No. 1, 1993, pp. 1–32.
doi:10.1006/imms.1993.1051
[47] Bolton, M. L., and Bass, E. J., "Evaluating Human–Automation Interaction Using Task Analytic Behavior Models, Strategic Knowledge-Based Erroneous Human Behavior Generation, and Model Checking," *Proceedings of the IEEE International Conference on Systems Man and Cybernetics*, IEEE, Piscataway, NJ, 2011, pp. 1788–1794.
[48] Fields, R. E., "Analysis of Erroneous Actions in the Design of Critical Systems," Ph.D. Thesis, Univ. of York, York, England, U.K., 2001.
[49] Paternò, F., and Santoro, C., "Preventing User Errors by Systematic Analysis of Deviations from the System Task Model," *International Journal of Human-Computer Studies*, Vol. 56, No. 2, 2002, pp. 225–245.
doi:10.1006/ijhc.2001.0523
[50] Bastide, R., and Basnyat, S., "Error Patterns: Systematic Investigation of Deviations in Task Models," *Task Models and Diagrams for Users Interface Design*, Springer, Berlin, 2007, pp. 109–121.
[51] Kieras, D. E., Wood, S. D., and Meyer, D. E., "Predictive Engineering Models Based on the EPIC Architecture for a Multimodal High-Performance Human-Computer Interaction Task," *ACM Transactions on Computer-Human Interaction*, Vol. 4, No. 3, 1997, pp. 230–275.
doi:10.1145/264645.264658
[52] John, B. E., and Kieras, D. E., "The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast," *ACM Transactions on Computer-Human Interaction*, Vol. 3, No. 4, 1996, pp. 320–351.
doi:10.1145/235833.236054

[53] Hollan, J., Hutchins, E., and Kirsh, D., "Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research," *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 2, 2000, pp. 174–196.
doi:10.1145/353485.353487

[54] Pritchett, A. R., Kim, S. Y., and Feigh, K. M., "Modeling Human–Automation Function Allocation," *Journal of Cognitive Engineering and Decision Making*, Vol. 8, No. 1, 2014, pp. 33–51.
doi:10.1177/1555343413490944

K. Feigh
Associate Editor