Evaluating Human-Human Communication Protocols with Miscommunication Generation and Model Checking

Matthew L. Bolton[1] and Ellen J. Bass[2]

[1] Department of Mechanical and Industrial Engineering
University of Illinois at Chicago, Chicago IL 60607
mbolton@uic.edu
[2] College of Information Science and Technology
College of Nursing and Health Professions
Drexel University, Philadelphia, PA 19104
Ellen.J.Bass@Drexel.edu

Abstract. Human-human communication is critical to safe operations in domains such as air transportation where airlines develop and train pilots on communication procedures with the goal to ensure that they check that verbal air traffic clearances are correctly heard and executed. Such communication protocols should be designed to be robust to miscommunication. However, they can fail in ways unanticipated by designers. In this work, we present a method for modeling human-human communication protocols using the Enhanced Operator Function Model with Communications (EOFMC), a task analytic modeling formalism that can be interpreted by a model checker. We describe how miscommunications can be generated from instantiated EOFMC models of human-human communication protocols. Using an air transportation example, we show how model checking can be used to evaluate if a given protocol will ensure successful communication. Avenues of future research are explored.

Keywords: Task analysis, Human-human communication, Air traffic control, Formal methods, Model checking, Human error.

1 Introduction

Human-human communication is critical to the safe operations of many complex systems. For example, until the Data Communication Integrated Services (DCIS) contract is fully implemented [25], voice communications will continue to be the primary mechanism for air traffic control clearances in the United States [1]. In many work domains, institutions develop human-human communication protocols to support safer operations. In air transportation, for example, the pilot/controller communication loop, using readbacks and other confirmation behaviors, is designed to support safety and redundancy of pilot/controller communications [32]. However miscommunications continue to impact the safety of complex systems. The Aviation Safety Reporting System (ASRS) data base, for example, identifies problems including incorrect communications, incomplete or absent communications, and correct but late communications [35].

G. Brat, N. Rungta, and A. Venet (Eds.): NFM 2013, LNCS 7871, pp. 48–62, 2013.

Further, Jones [32] has identified a number of air transportation accidents where miscommunication played a significant role.

As miscommunications continue, institutions will try to enhance human-human communication protocols. Thus having methods to verify such protocols can improve safety. There is a long tradition of formal methods being used to describe and formally verify machine communication protocols [2, 8, 19, 22, 38, 42] including the injection of communication faults or errors into the models [21, 41]. While human-human communication protocols could be modeled using these traditional formal methods approaches, human-human communication, which can include verbal statements, gestures, and related actions, is different from machine communication. Human communications are actions [3] that occur as part of the participants' larger tasks (i.e., goal directed normative behaviors to accomplish system goals [34]).

Few researchers have used formal methods to evaluate human-human communication protocols. Hörl and Aichernig [29, 30] developed a formal model of the system 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. Thus, Hörl and Aichernig avoided having to explicitly model and evaluate protocols with a task by having the actual tests provide that context. Others have investigated how task analytic models can be incorporated into formal models and evaluated with formal verification (see [17] for a review). Only Paternò et al. [37] and Bass et al. [4] treat human-human communication as actions [4] within a larger set of coordinated communication and task relevant activities. With the Enhanced Operator Function Model with Communications (EOFMC), Bass et al. [4] introduced an innovative means of representing goal directed behaviors requiring 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 [37] and [4] have investigated how miscommunications could result in the failure of human-human communication protocols.

1.1 Objectives

Methods are needed to support formal evaluation of human-human communication protocols, including potential miscommunications. In this work, we describe a novel approach that allows an analyst to model human-human communication protocols in the context of a task analytic modeling formalism and to use formal verification with model checking to evaluate whether or not the protocol will always ensure that a correct communication will occur, even with miscommunication. In the following sections, we describe our method and its implementation. We present EOMFC, the task analytic modeling formalism we use for modeling protocols; the process that is used to translate instantiated EOFMCs into the input language of a model checker; and the modification to the translation process that allows the formal representation to be capable of generating miscommunications. We also present an air traffic control application to illustrate

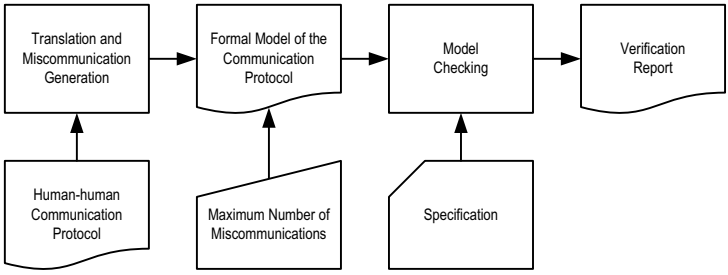

Fig. 1. Human-human communication protocol analysis method

how our method can discover problems with safety critical, human-human communication protocols. Finally, we discuss our results and outline avenues of future research.

2 Method

The method in Fig. 1 was extended from [11, 15] 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 which 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 which 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 [19]. Model checking produces a verification report either confirming that the model adheres to the specification or a counterexample, illustrating how the specification was violated.

2.1 Human-Human Communication Protocol Modeling

To model human-human communication protocols, we use the Enhanced Operator Function Model with Communications [4] (an extension of the Enhanced Operator Function Model (EOFM) [9, 10, 18]). EOFMCs, with their formal semantics, are task analytic modeling formalisms capable of representing multiple human operators and human-human communication as part of a larger task model. They allow communication protocols to be modeled as shared task structures on their own or with other tasks.

EOFMC models groups of human operators engaging in shared activities as an input/output system. Inputs may come from the human-device 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 and 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 is 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: (a) they can be observable, singular ways the human operator can interact with the environment; (b) they can represent a cognitive or perceptual act, where a value is assigned to a local variable; (c) 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). EOFMC supports ten different decomposition operators [4]. Herein, only the following are used:

1. *and_par* – all activities or action in the decomposition must execute (in any order) and their execution can overlap;
2. *ord* – all activities or actions in the decomposition must execute in order; and
3. *com* – 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 tree-like graph (such as Fig. 5 on page 56) where actions are depicted by rectangular nodes and activities by rounded rectangle nodes. Conditions are connected to the activity they modify: a *precondition* is represented by a yellow, downward pointing triangle connected to the right side of the activity; a *completioncondition* is presented as a magenta, upward pointing triangle connected to the left of the activity; and a *repeatcondition* is conveyed as a recursive arrow attached to the top of the activity. These standard colors are used to distinguish condition shapes from each other and other task structures. Decompositions are arrows, labeled with the decomposition operator, extending below an activity pointing to a large rounded rectangle with the decomposed activities or actions.

By exploiting the shared activity and communication action feature of EOFMC, human-human communication protocols can be modeled as shared task activities. Human communication actions can represent human-human communication. However, other actions 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.

2.2 EOFMC Formal Semantics and Translation

EOFMC has formal semantics which specify how an instantiated EOFMC model executes. Each activity or action has one of three execution states: waiting to execute (*Ready*), executing (*Executing*), and done (*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 [18] for more details.

Instantiated EOFMC task models can be translated into the language of the Symbolic Analysis Laboratory (SAL) [20] (in this case using a java program) using the EOFMC formal semantics in virtually the same manner as EOFMs [18]. The major difference between EOFMC and EOFM translation is how communications are handled – how actions in a *com* decomposition (not present in EOFM) 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*; 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 (Fig. 2 shows the SAL notation).

The translated EOFMC can be integrated into a larger system model using a defined architecture and coordination protocol [12, 18]. Formal verifications are performed using SAL's Symbolic Model Checker (SAL-SMC). Any produced counterexamples can be visualized and evaluated using EOFMC's visual notation (extended from [13]).

```
[]StartCondition -->
    HumanComAction'       = Done;
    ComActionValue'       = ComValue;
    LocalVariableAction1' = Done;
    LocalVariable1'       = ComActionValue';
    ...
    LocalVariableActionN' = Done;
    LocalVariableValueN'  = ComActionValue';
```

Fig. 2. Pattern of code generated to represent a human-human communication in the SAL code (see [20]) created from translating an instantiated EOFMC. `[]StartCondition -->` represents a nondeterministic guarded transition. 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.

2.3 Miscommunication Generation

There are many reasons why human-human miscommunication can occur (see [24]). From an engineering and design perspective, a miscommunication can be viewed as 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 [43]. To support miscommunication generation, the translator was modified to include an additional optional transition for each original communication transition (Fig. 3).

```
[]StartCondition AND (ComErrorCount < ComErrorMax) -->
    HumanComAction'       =  Done;
    ComActionValue'       IN {x: CommunicationType | TRUE};
    LocalVariableAction1' =  Done;
    LocalVariable1'       IN {x: CommunicationType | TRUE};
    ...
    LocalVariableActionN' =  Done;
    LocalVariableN'       IN {x: CommunicationType | TRUE};
    ComErrorCount'        =  IF   ComActionValue' <> ComValue
                                  OR LocalVariable1' <> ComValue
                                  OR ...
                                  OR LocalVariableN' <> ComValue
                             THEN ComErrorCount + 1
                             ELSE ComErrorCount ENDIF;
```

Fig. 3. Pattern of code generated to represent a human-human miscommunication in the SAL code created from translating an instantiated EOFMC. Notation is as described in Fig. 2 with several additions. First, `IN` is used to indicate that the variable to the left of it can assume any value in the set to the right. Second, {`x: CommunicationType | TRUE`} indicates a set containing all the elements defined in type of the communication value (`CommunicationType`). Further, `ComErrorCount` represents the total number of miscommunications that have occurred and `ComErrorMax` represents the total number of miscommunications that are allowed to occur. Finally, the `IF ... THEN ... ELSE ... ENDIF` statement only allows `ComErrorCount` to be incremented if a miscommunication has occurred.

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. 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 (`ComErrorCount`) to track the number of miscommunications. The transition ensures that a miscommunication transition can only occur if maximum has not been reached.

3 Application

To illustrate how our approach can model a safety critical human-human communication protocol, we construct an instantiated EOFMC model for an aircraft heading

change in an air transportation example and we use our method to evaluate whether or not it is robust for up to one miscommunication. This example has three human operators, two pilots (a pilot flying and a pilot monitoring) and an air traffic controller. In the scenario, an air traffic controller wants to clear the aircraft to a new heading.

Both the pilots and the air traffic controller have push-to-talk switches which they press down when they want to verbally communicate information to each other over the radio. They can release this switch to end communication.

With respect to the aircraft, the Autopilot Flight Director System consists of Flight Control Computers and the Mode Control Panel (MCP). The MCP provides control of the Autopilot (A/P), Flight Director, and the Autothrottle System. When the A/P is engaged, the MCP sends commands to the aircraft pitch and roll servos to operate the aircraft flight control surfaces. Herein the MCP is used to activate heading changes. The Heading (HDG)/Tracking (TRK) window of the MCP displays the selected heading or track (Fig. 4). The 3 digit numeric display provides the current desired heading in compass degrees (between 0 and 359). Below the HDG window is the heading select knob. Changes in the heading are achieved by rotating and pulling the knob. Pulling the knob tells the autopilot to use the pilot selected value and engages the HDG mode.

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

1. The air traffic controller contacts the pilots and gives them a new heading clearance.
2. The pilot monitoring re-contacts air traffic control and repeats the heading.
3. If the heading read back to the air traffic controller is not the heading that the air traffic controller intended, then this process needs to be repeated (starting at step 1) until the correct heading is read back.
4. Next, the pilot flying goes through the process of entering the new heading.
5. Before engaging the new heading, the pilot monitoring points at the heading window and reads off the entered heading.
6. If the heading read back by the pilot monitoring does not match the heading that the pilot monitoring hears from air traffic control, he must then repeat the process for entering and confirming the heading (going back to step 4).
7. The pilot engages the entered heading.

We next show how we can instantiate this protocol in an EOFMC and use formal verification to prove whether it will always ensure that the correct heading is engaged.

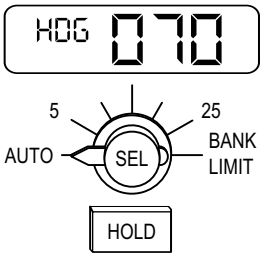

Fig. 4. Heading control and display

3.1 Modeling

This communication protocol was implemented as an instantiated EOFMC (Fig. 5). This model has three human operators: the air traffic controller (ATCo), the pilot flying (PF), and the pilot monitoring (PM). The entire process starts when the air traffic controller presses his push-to-talk switch. Then, the controller communicates the heading (*lATCoSelectedClearance*[1]) to the pilots (via the *hATCoTalk* human communication action), such as "AC1 Heading 070 for spacing." Both pilots remember this heading (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. The PM then repeats/communicates the heading that he heard (in this example, "AC1 Heading 070"), where both the ATCo and PF hear and remember the heading. The PM releases the switch. This entire process must repeat if the heading the ATCo hears from the PM does not match the heading he wanted to communicate (*lATCSelectedClearance* $\neq$ *lATCHeadingHeardFromPilots*). It completes otherwise.

Once the heading has been communicated, the pilots collaborate to set the new heading (*aSetNewHeading*). This process involves selecting and confirming the heading (*aChangeAndConfirm*) and then executing the new heading (*aExecuteTheChange*). The selection and confirmation process 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 selection in the window and stating the entered heading. Here, two communications occur in parallel (indicated by the *and_par* decomposition operator associated with *aConfirmTheChange*): the PM points at the heading window (*aPointAtHeadingWindow*) and he speaks the heading that was entered (*aSayTheHeading*). Both are perceived by the PF. This process must repeat if the heading the PF perceived from the ATCo does not match the heading spoken by the PM (*lPFHeadingFromPM* $\neq$ *lPFHeadingFromATCo*). Once the heading is confirmed, the PF presses the heading select (hold) button to execute the heading change.

3.2 Translation

The instantiated EOFMC was translated into SAL using the automated translator. The original model contained 144 lines of XML code. The translated model contained 404 lines of SAL code. This model was composed with one representing the heading change window, where the heading can be changed when the pilot rotates the heading knob. These two model were 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.

[1] Note that in this example, all headings are modeled abstractly as either being *CorrectHeading*, if it matches the heading clearance the ATCo intended to communicate, or *IncorrectHeading*, if it does not.

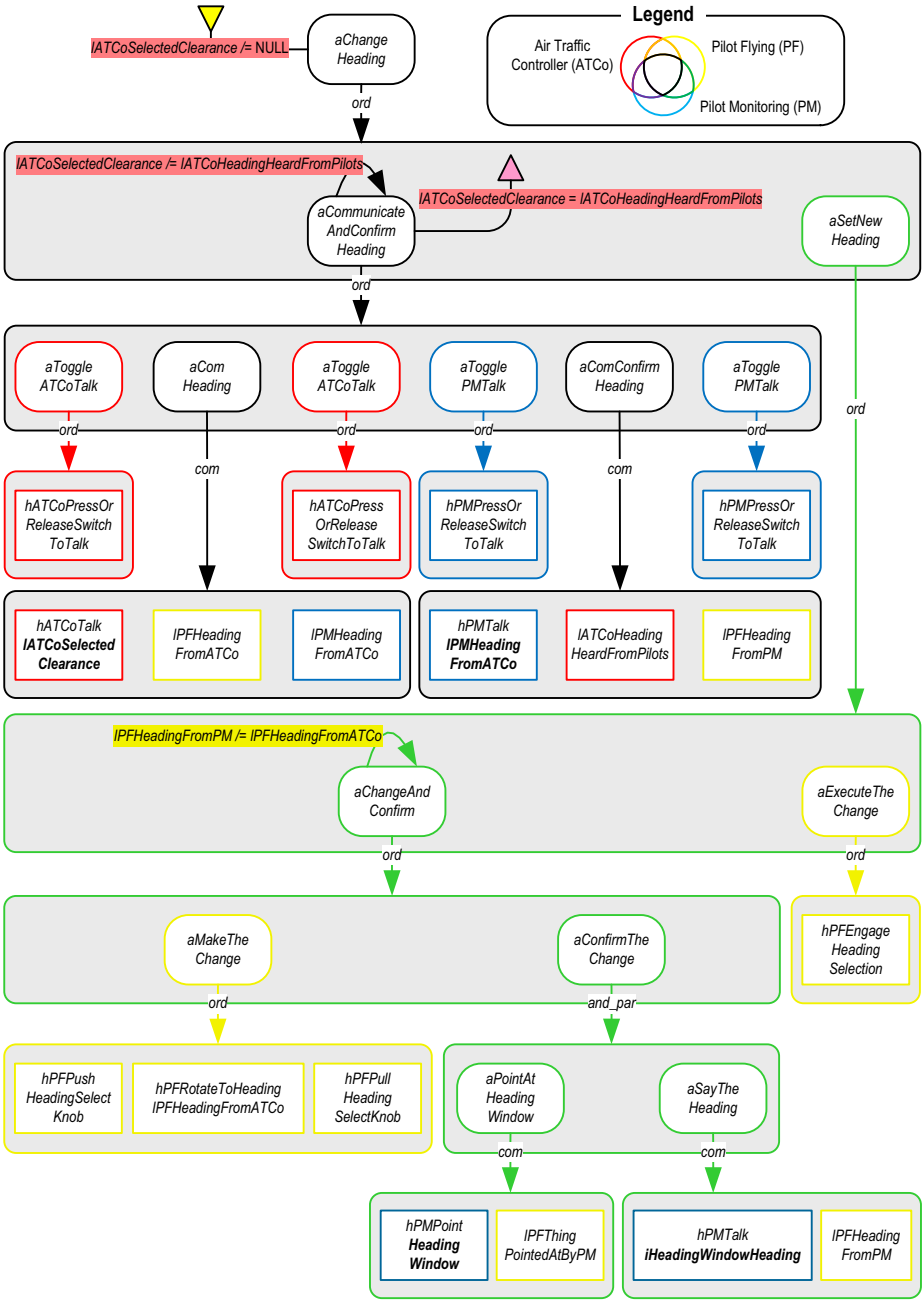

Fig. 5. Visualization of the instantiated EOFMC communication protocol for changing the aircraft heading. Activities are prefixed by "*a*", actions by "*h*", inputs by "*i*", and local variables by "*l*". Values or variables used in a communication action are bolded.

3.3 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, we can formulate this into linear temporal logic (the specification logic used by SAL) as follows:

$$\mathbf{G}\begin{pmatrix}(aChangeHeading = Done)\\ \rightarrow (iHeadingWindowHeading = lATCSelectedClearance)\end{pmatrix} \quad (1)$$

This specification was checked against the two versions of the formal model using SAL's symbolic model checker (sal-smc) on a workstation with 16 gigabytes of RAM, a 3.0 gigahertz dual-core Intel Xeon processor, and the Ubuntu 9.04 desktop.

The first model (`ComErrorMax = 0`), verified to true in 2.52 seconds (total execution time) having visited 559 states.[2]. The second model (`ComErrorMax = 1`) returned a counterexample after 3.2 seconds (total execution time) having visited 3726 states.

3.4 Failure Diagnosis

To help diagnose why this failure occurred, the counterexample was visualized using the technique described in [13]. This revealed the following failure sequence:

- The air traffic controller, wanting to clear the aircraft to a new heading (*CorrectHeading*), presses the switch to talk.
- The air traffic controller issues a clearance to *CorrectHeading*. However, a miscommunication occurs and the pilot flying thinks he heard *IncorrectHeading*.
- The air traffic controller releases the switch.
- The pilot monitoring presses his switch to talk.
- The pilot monitoring repeats back the heading he heard from the air traffic controller without a miscommunication occurring.
- The pilot monitoring releases the switch to talk.
- Since the correct heading was heard by the pilot monitoring, the air traffic controller allows the activity for communicating the heading (*aCommunicateAndConfirmHeading*) to complete its execution.
- The pilots begin collaborating to enter the heading from the air traffic controller.
- The pilot flying performs the activity for changing the heading: he presses the heading select knob, sets the dial to the heading he heard from the air traffic controller (*IncorrectHeading*), and pulls the heading select knob.
- The pilot monitoring then pointed at the heading display and read off the heading entered (*IncorrectHeading*) to the pilot flying.
- Because the heading just heard from the pilot monitoring (*lPFHeadingFromPM2*) matches the heading the pilot flying heard from air traffic control (*lPFHeadingFromATC*), the pilot flying engages the new heading.

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

[2] Note that an additional verification was conducted using the specification $\mathbf{F}(aChangeHeading = Done)$ to ensure that (1) was not true do to vacuity.

4 Discussion and Future Work

Human-human communication protocols can be critical to the safe operation of a system, and can fail in unexpected ways. Herein, we have 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 non-verbal information and result in non-communication human actions, this work considers human communication as part of human task behavior. We used EOFMC 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. We described the EOFMC modeling language, its formal semantics, and the process used to translate the formal models into a model checking language. We introduced a new method for automatically generating miscommunications between human operators and showed how these could be automatically included in the translated representation of an instantiated EOFMC communication protocol. We also presented an air traffic control application to demonstrate how this method could be used to find problems in a human-human communication protocol for a safety critical system.

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

4.1 Design Interventions and Additional Analyses

The failure discovered in the presented human-human communication protocol appears to occur because the protocol does not give the two pilots a means of reconciling differences between the headings they heard from air traffic control. Thus, potential solutions should support both coordinated error detection and recovery. There may be a number of ways to accomplish this. Two possible approaches are highlighted here. Firstly, if there is any disagreement between what the two pilots heard from the air traffic controller, then they could consult the air traffic controller to reconcile the disagreement. However, doing this could add additional work to the already busy air traffic controller (a potentially undesirable strategy). Alternatively, the pilots could reconcile among each other to determine what the original air traffic controller's clearance was. To determine which of these solutions is most effective, they would need to be encoded into new human-human communication protocols and evaluated with our presented method. Future work should perform these analyses.

Further, the analyses presented here only considered a maximum of one miscommunication. Ideally, a human-human communication protocol would be robust to more than just one. Thus, future work should investigate whether candidate protocols are robust for `ComErrorMax > 1`.

4.2 Scalability

The more than six times increase in state space size observed in the model checking of the model with no miscommunications (`ComErrorMax = 0`) and the one with up to one

miscommunication (`ComErrorMax = 1`) suggests that there could be scalability problems with the presented method. Further, scalability assessments of formal verification analyses that have included task analytic models suggest that the state space size can increase exponentially with the size of the task model [16]. These factors could limit what types of human-human communication protocols our method could be used to evaluate and/or the maximum number of miscommunications that could be considered in a given verification. Future work should evaluate how this method scales, identify what factors most influence its scalability, and determine what types of human-human communication protocols can be evaluated using it without running into scalability problems.

4.3 Other Modeling Formalisms

SAL was used in this work because EOFM, the base task analytic modeling formalism, uses SAL. This made it easy to adapt the existing translation tools for use in EOFMC. However, it is possible that other formal tools could prove to be better suited to this particular application. For example, Communicating Sequential Processes (CSP) [26] natively models communication protocols and thus may be better suited to this work. Other formal modeling languages and tools could conceivably help address the scalability concerns discussed above. Other formal modeling infrastructures should be considered in future work.

4.4 Miscommunication Extensions

When generating miscommunications using our 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 [24]. 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 our 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.

4.5 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 [7, 28, 40]. Related work has investigated how to generate erroneous human behavior in task analytic models and evaluate its impact on systems using model checking [14, 16]. Thus, it should be possible to evaluate how robust human-human communication protocols are to these other types of erroneous human behavior.

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 [6, 14, 16, 23, 36]. Human-human communication protocols can have each of the human operators doing specific elements of a task on his or her own, but also requires 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.

4.6 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 [5, 27, 31, 33, 39]. Future work should compare our method with these, determine what the tradeoffs are between them, and investigate possible avenues of synergy.

Acknowledgement. The research was funded by NASA Ames Research Center award NNA10DE79C: NextGenAA: Integrated model checking and simulation of NextGen authority and autonomy.

References

1. Airbus: Effective pilot/controller communications. In: Human Performance. Flight Operations Briefing Notes. Airbus, Blagnac Cedex (2006)
2. Argón, P., Delzanno, G., Mukhopadhyay, S., Podelski, A.: Model checking communication protocols. In: Pacholski, L., Ružička, P. (eds.) SOFSEM 2001. LNCS, vol. 2234, pp. 160–170. Springer, Heidelberg (2001)
3. Austin, J.: How to do things with words, vol. 88. Harvard University Press (1975)
4. Bass, E.J., Bolton, M.L., Feigh, K., Griffith, D., Gunter, E., Mansky, W., Rushby, J.: Toward a multi-method approach to formalizing human-automation interaction and human-human communications. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 1817–1824. IEEE, Piscataway (2011)
5. Bass, E.J., Baxter, G.D., Ritter, F.E.: Creating models to control simulations: A generic approach. AI and Simulation of Behaviour Quarterly 93, 18–25 (1995)
6. Bastide, R., Basnyat, S.: Error patterns: Systematic investigation of deviations in task models. In: Coninx, K., Luyten, K., Schneider, K.A. (eds.) TAMODIA 2006. LNCS, vol. 4385, pp. 109–121. Springer, Heidelberg (2007)
7. Baxter, G.D., Bass, E.J.: Human error revisited: Some lessons for situation awareness. In: Proceedings of the Fourth Annual Symposium on Human Interaction with Complex Systems, pp. 81–87. IEEE (1998)
8. Bochmann, G., Sunshine, C.: Formal methods in communication protocol design. IEEE Transactions on Communications 28(4), 624–631 (1980)
9. Bolton, M.L.: Automatic validation and failure diagnosis of human-device interfaces using task analytic models and model checking. Computational and Mathematical Organization Theory, 1–25 (2012), http://dx.doi.org/10.1007/s10588-012-9138-6
10. Bolton, M.L., Bass, E.J.: Enhanced operator function model: A generic human task behavior modeling language. In: Proceedings of the IEEE International Conference on Systems Man and Cybernetics, pp. 2983–2990. IEEE, Piscataway (2009)

11. Bolton, M.L., Bass, E.J.: A method for the formal verification of human interactive systems. In: Proceedings of the 53rd Annual Meeting of the Human Factors and Ergonomics Society, pp. 764–768. HFES, Santa Monica (2009)
12. Bolton, M.L., 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 6(3), 219–231 (2010)
13. Bolton, M.L., Bass, E.J.: Using task analytic models to visualize model checker counterexamples. In: Proceedings of the 2010 IEEE International Conference on Systems, Man, and Cybernetics, pp. 2069–2074. IEEE, Piscataway (2010)
14. Bolton, M.L., Bass, E.J.: Evaluating human-automation interaction using task analytic behavior models, strategic knowledge-based erroneous human behavior generation, and model checking. In: Proceedings of the IEEE International Conference on Systems Man and Cybernetics, pp. 1788–1794. IEEE, Piscataway (2011)
15. Bolton, M.L., Bass, E.J.: Using model checking to explore checklist-guided pilot behavior. International Journal of Aviation Psychology 22, 343–366 (2012)
16. Bolton, M.L., Bass, E.J., Siminiceanu, R.I.: Using phenotypical erroneous human behavior generation to evaluate human-automation interaction using model checking. International Journal of Human-Computer Studies 70, 888–906 (2012)
17. Bolton, M.L., Bass, E.J., 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 (in press, expected 2013)
18. Bolton, M.L., Siminiceanu, R.I., 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), 961–976 (2011)
19. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press, Cambridge (1999)
20. De Moura, L., Owre, S., Shankar, N.: The SAL language manual. Tech. Rep. CSL-01-01, Computer Science Laboratory, SRI International, Menlo Park (2003)
21. Dietrich, F., Hubaux, J.: Formal methods for communication services. Tech. Rep. SSC/1999/023, Institute for Computer Communications and Applications, Swiss Federal Institute of Technology (1999)
22. Edelkamp, S., Leue, S., Lluch-Lafuente, A.: Directed explicit-state model checking in the validation of communication protocols. International Journal on Software Tools for Technology Transfer 5(2), 247–267 (2004)
23. Fields, R.E.: Analysis of Erroneous Actions in the Design of Critical Systems. Ph.D. thesis, University of York, York (2001)
24. Gibson, W., Megaw, E., Young, M., Lowe, E.: A taxonomy of human communication errors and application to railway track maintenance. Cognition, Technology & Work 8(1), 57–66 (2006)
25. Harris Corporation: Harris Corporation awarded $331 million contract by FAA for data communications integrated services program (2012), `http://harris.com/view_pressrelease.asp?act=lookup&pr_id=3518` (accessed December 16, 2012)
26. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM 21(8), 666–677 (1978)
27. Hollan, J., Hutchins, E., Kirsh, D.: Distributed cognition: toward a new foundation for human-computer interaction research. ACM Transactions on Computer-Human Interaction 7(2), 174–196 (2000)
28. Hollnagel, E.: The phenotype of erroneous actions. International Journal of Man-Machine Studies 39(1), 1–32 (1993)
29. Hörl, J., 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++. In: Wing, J.M., Woodcock, J., Davies, J. (eds.) FM 1999. LNCS, vol. 1709, pp. 1868–1868. Springer, Heidelberg (1999)

30. Hörl, J., Aichernig, B.K.: Validating voice communication requirements using lightweight formal methods. IEEE Software 17(3), 21–27 (2000)
31. John, B.E., Kieras, D.E.: The goms family of user interface analysis techniques: comparison and contrast. ACM Transactions on Computer-Human Interaction 3(4), 320–351 (1996)
32. Jones, R.K.: Miscommunication between pilots and air traffic control. Language Problems and Language Planning 27(3), 233–248 (2003)
33. Kieras, D.E., Wood, S.D., 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 4(3), 230–275 (1997)
34. Kirwan, B., Ainsworth, L.K.: A Guide to Task Analysis. Taylor and Francis, London (1992)
35. NASA Aviation Safety Reporting System: Pilot/controller communications. Tech. rep., NASA Ames Research Center (2012)
36. Paternò, F., Santoro, C.: Preventing user errors by systematic analysis of deviations from the system task model. International Journal of Human-Computer Studies 56(2), 225–245 (2002)
37. Paternò, F., Santoro, C., Tahmassebi, S.: Formal model for cooperative tasks: Concepts and an application for en-route air traffic control. In: Proceedings of the 5th International Conference on the Design, Specification, and Verification of Interactive Systems, pp. 71–86. Springer, Vienna (1998)
38. Pek, E., Bogunovic, N.: Formal verification of communication protocols in distributed systems. In: Proceedings of MIPRO 2003, Computers in Technical Systems and Intelligent Systems, pp. 44–49. MIPRO (2003)
39. Pritchett, A.R., Feigh, K.M., Kim, S.Y., Kannan, S.: Work models that compute to support the design of multi-agent socio-technical systems (under review)
40. Reason, J.: Human Error. Cambridge University Press, New York (1990)
41. Sidhu, D.P., Leung, T.: Formal methods for protocol testing: A detailed study. IEEE Transactions on Software Engineering 15(4), 413–426 (1989)
42. Sunshine, C.A.: Formal methods for communication protocol specification and verification. Tech. rep., RAND Corporation, Santa Monica (1979)
43. Traum, D., Dillenbourg, P.: Miscommunication in multi-modal collaboration. In: AAAI Workshop on Detecting, Repairing, and Preventing Human–Machine Miscommunication, pp. 37–46. AAAI, Palo Alto (1996)