Using Formal Verification to Evaluate Human-Automation Interaction: A Review

Matthew L. Bolton, *Member, IEEE*, Ellen J. Bass, *Senior Member, IEEE*, and Radu I. Siminiceanu

***Abstract*—Failures in complex systems controlled by human operators can be difficult to anticipate because of unexpected interactions between the elements that compose the system, including human-automation interaction (HAI). HAI analyses would benefit from techniques that support investigating the possible combinations of system conditions and HAIs that might result in failures. Formal verification is a powerful technique used to mathematically prove that an appropriately scaled model of a system does or does not exhibit desirable properties. This paper discusses how formal verification has been used to evaluate HAI. It has been used to evaluate human-automation interfaces for usability properties and to find potential mode confusion. It has also been used to evaluate system safety properties in light of formally modeled task analytic human behavior. While capable of providing insights into problems associated with HAI, formal verification does not scale as well as other techniques such as simulation. However, advances in formal verification continue to address this problem, and approaches that allow it to complement more traditional analysis methods can potentially avoid this limitation.**

***Index Terms*—Cognitive task analysis, formal methods, human performance modeling, human-automation interaction (HAI), mode confusion, model checking, theorem proving, verification.**

I. Introduction

THE DESIGN of human-automation interaction (HAI) has contributed to failures in domains including aviation [1]–[4], process control [5], and medicine [6]. For example, HAI has played an important role in the crashes of American Airlines Flight 965 [7] and China Air 140 [8]; the grounding of the Royal Majesty cruise ship [9]; the disaster at Three Mile Island [10]; and the death of medical patients with the Therac-25 [11]. Despite improvements in HAI design practices, it is difficult to anticipate all potential interactions within and between system components (human operators, human-automation interfaces, device automation, and conditions in the operational environment) [10], [12].

A. HAI

Researchers have identified factors that contribute to problems with HAI along with technologies and tools to address these problems [13], [14].

1) Sources of HAI Failures: Problems with HAI can occur for a number of reasons. Automation may be brittle, as it can only respond to the pre-specified situations for which it was designed [15]. With brittle automation, operators may encounter problems in situations outside the scope of the automation's design parameters [15]–[18]. The human operator may also find it difficult to combine cues and decision support provided by automation with additional information not used by automation, but deemed important by the operator [18]. Unfortunately, a system may not identify when the limits of its automation are reached. HAI design paradigms have been developed to address brittle automation. For example, end users may be allowed to customize automation [19] or the automated systems may be allowed to expand their capabilities after being fielded [20]. However, problems can still arise because operational conditions or device behaviors were not anticipated by the designer; the design of automation was oversimplified due to schedule, economic, or technological limitations; or the device automation and/or human-automation interface were not implemented in accordance with the design.

Human-automation interfaces may not provide enough feedback about the state of the device automation [21], [22]. Further, human operators may not properly understand how the automation works (they may not have a correct mental model) [23]. Both of these conditions can lead to mode confusion, where the human operator is unable to keep track of the state or mode of the device automation. This is dangerous because mode confusion can result in automation surprise, where the operator's situation awareness is disrupted by the device automation behaving in an unexpected way [23], [24]. Further, mode confusion can lead to the human operator either performing inappropriate actions (errors of commission) or omitting necessary ones (errors of omission) [23]. Thus, operators of automated systems must work to maintain mode awareness to avoid such errors [25]. This may be very difficult given the large number of mode combinations, the variety of behaviors a given mode can exhibit, and the range and quantity of information displayed in some systems [23].

As systems become more automated, the tasks of the human operator change. For example, he or she may need to supervise and monitor the operation of the system rather than control it

Manuscript received November 9, 2010; revised August 8, 2011 and March 14, 2012; accepted June 24, 2012. The work documented here was completed while Matthew L. Bolton was a Senior Research Associate for San José State University Research Foundation at NASA Ames Research Center. The project was supported in part by Grant Number T15LM009462 from the National Library of Medicine (NLM), NASA Cooperative Agreement NCC1002043, NASA Contract NNA10DE79C, and FAA Task Number 09-AJP61SSP-0071. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NLM, NASA, NIA, FAA, or the National Institutes of Health. This paper was recommended by Associate Editor C. Cao.

M. L. Bolton is with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA (e-mail: mbolton@uic.edu).

E. J. Bass is with the College of Information Science and Technology and College of Nursing and Health Professions, Drexel University, Philadelphia, PA 19104 USA.

R. I. Siminiceanu is with the National Institute of Aerospace, Hampton, VA 23666, USA.

Digital Object Identifier 10.1109/TSMCA.2012.2210406

directly. In such situations, human operators may not have been trained in these new tasks [26]. They may fixate on a specific task while neglecting others, such as passively monitoring the system [15], [27]–[30]. They may also alter their task behavior such that they become too dependent on automation (a condition known as automation bias) and therefore acquire and evaluate system information less vigilantly [17], [31].

Problems can also arise as a result of incompatibilities between the behavior of the device (both its automation and human-automation interface) and the cognition of the human operator. Many erroneous human behaviors have predictable cognitive causes [12] related to human working memory, human knowledge, human perception, or human physical coordination. It is possible to prevent some erroneous human behaviors through changes to the behavior of the device's automation and human-automation interface [32]–[35].

Thus, problems with HAI can arise as a result of inadequacies in the human-automation interface (due to problems of usability or mode) or more system-level problems related to unanticipated interactions between the human operator's behavior and the system (related to the human task, other cognitive reasons, unanticipated limitations of the automation, unforseen operational conditions, or some combination).

2) Designing for and Evaluating HAI: These problems have been addressed from different directions. Techniques such as cognitive work analysis [36] and cognitive task analysis [37], [38] have been developed. Cognitive work analysis is concerned with identifying constraints in the operational environment that shape the mission goals of the human operator and can be used to characterize automation (see, for example, [39]). Cognitive task analysis is concerned with describing how (physically and/or cognitively) human operators normatively and descriptively perform goal-oriented tasks when interacting with an automated system [40]–[42]. These techniques influence the design of human-automation interfaces by identifying device and environmental information to faciliate human interpretation and the tasks to achieve system goals [43]–[45]. Techniques involving human subject experiments allow analysts to evaluate the effectiveness of different human-automation interactive systems using a variety of metrics (see [46]–[50]) in real operational environments and part task simulations. Finally, a number of cognitive, decision making, behavioral classification, and modeling techniques have been developed to give analysts means of performing manual, simulation, and statistical evaluations without the need for human subject experiments (see [13], [51], [52]).

While these techniques are useful in finding and/or potentially reducing the likelihood of HAI-related system failures, they are not exhaustive and thus risk missing potentially dangerous HAIs that could lead to system failures. Verification techniques that fall under the banner of formal methods offer means of performing appropriately scaled exhaustive analyses that can augment existing methods. Sections II and III highlight the advantages of such analyses.

B. Formal Verification

Formal verification is an analysis technique that falls in the broader category of formal methods. Formal methods are a set of well-defined mathematical languages and techniques for the specification, modeling, and verification of systems [53]. Specifications are formulated to describe desirable system properties in rigorous, unambiguous notations. Systems are modeled using mathematically based languages that support well established theoretical formalisms such as finite state automata, directed graphs, Büchi automata, Petri nets, and μ-calculus. The verification process mathematically proves whether the model satisfies the specification. Formal verification has been used successfully in a number of applications, particularly computer hardware, where performance must be guaranteed. Two particular technologies, automated theorem proving and model checking, have been useful for the formal verification of large systems.

1) Automated Theorem Proving: Theorem proving is a deductive technique that resembles traditional pencil-and-paper proofs: from a set of axioms, using a set inference rules, one builds theories and proves theorems to verify correctness claims about the system under investigation with the help of a proof assistant program. While theorem proving cannot be fully automated in practice for more expressive logics (such as higher order logics), some smaller fragments are amenable to mechanized proofs. Satisfiability solvers [54] and satisfiability modulo theories solvers [55] include powerful decision procedures able to solve very complicated problems from areas such as propositional logic and linear algebra.

2) Model Checking: Model checking is a highly automated approach used to verify that a formal model of a system satisfies a set of desired properties (a specification) [56]. A formal model describes a system as a set of variables and transitions between variable states. Specification properties are usually represented in a temporal logic (discussed below) using the formal system model variables to construct propositions. Verification is performed automatically by exhaustively searching a system's statespace to determine if these propositions hold. If there is a violation, an execution trace called a counterexample is produced. This counterexample depicts a model state (the value of the model's variables) corresponding to a specification violation along with a list of the incremental model states leading to the violation.

A temporal logic is a set of rules and symbols that allows time to be expressed and reasoned about as part of a logical framework: time is represented by an ordinal sequence of states [57]. For model checking purposes, a temporal logic formula is composed of Boolean propositions about the model variables and modal operators. Modal operators usually specify the temporal relationships between propositions. The two most common temporal logics (linear temporal logic and computation tree logic) make use of the modal operators described in Table I. There are different specification languages and/or modal logics of which these operators represent the most common concepts. See [57] for more detail.

3) Limitations of Formal Verification: While the mathematical proofs offered by formal verification techniques are extremely powerful, their use is limited by a variety of factors. One of the major challenges facing model checking verification is the state explosion problem. As more elements are incorporated into the system model, the memory and time required to store the combinatorially expanding statespace can easily

TABLE I
FREQUENTLY USED TEMPORAL LOGIC OPERATORS

Operator Type	Name	Usage	Interpretation
Path Quantifier	**A**ll	**A** ψ	Starting from the current state, all future paths satisfy ψ.
	Exists	**E** ψ	Starting from the current state, there is at least one path that satisfies ψ.
Temporal Operator	Ne**X**t	**X** ψ	ψ is true in the next state of a given path.
	Future	**F** ψ	ψ is eventually true in some future state of a given path.
	Global	**G** ψ	ψ will always be true in a given path.
	Until	ϕ **U** ψ	ϕ will be true until ψ is true for a given path.

exceed the available resources. Symbolic model checking [58] addresses this by using extremely efficient means of representing a system's statespace. Other techniques allow select portions of the statespace to be searched without compromising the accuracy of the verification. The best known of these depend on partial order reduction [59], symmetry [60], and abstraction techniques such as abstract interpretation [61] and counterexample-guided abstraction refinement [62]. However, even with these efficient representations, model checking is still limited in terms of the size of systems it can be used to evaluate.

A second limitation of model checking is the expressive power of its modeling formalisms. Traditional model checking is applied to systems that can be modeled with discrete variables. However, systems can have continuous quantities. While the field of hybrid systems has been developed to address this issue [63], current techniques can handle systems models with no more than a half-dozen continuous variables. With respect to the modeling of time, discrete-state models can be augmented with a set of clocks [64]. While this technique can be used to model clock synchronization problems in digital systems, only more simple models can be fully verified.

In principle, theorem proving does not suffer from the same limitations as model checking. However, theorem proving is not a fully automated process: the analyst guides the proof while exploiting automation to reuse routine proof techniques. The more expressive the logic used to model and reason about the system, the less automation is possible. Thus, theorem proving requires significant effort by highly trained experts who guide the verification process. Further, because the proof process must be guided, theorem proving is less likely to find emergent features that are not anticipated by the analyst.

A problem faced by all forms of formal verification is that of model validity. To provide useful insights about actual systems, the models used in formal verification need to be valid. If they are not, the verification process has the potential to find problems that may not exist in the actual system, or miss problems that exist in the actual system but not the models.

C. Formal Verification of HAI

Researchers have used formal verification to evaluate HAI. These techniques focus on abstract models from the HAI literature that can be represented with discrete mathematical models and used in analyses of a scope such that specific HAI problems can be discovered. Prior analyses have investigated potential problems with human-automation interfaces related to both usability and mode confusion. They have also investigated how task-analytic or cognitively plausible human behavior may interact with other system automation or environmental conditions to result in violations of desirable system properties. We survey the literature as it relates to each of these topics. For each covered approach, we present the theory behind it and explain what types of problems it is meant to address. We briefly present an illustrative example from the literature that shows how the approach can be applied. We discuss the major developments and limitations of each approach. Finally, we discuss future ways in which formal verification (in general) might be extended to the evaluation of HAI and how these types of analyses may be used synergistically with more traditional HAI evaluation and analysis techniques.

II. FORMAL VERIFICATION OF HUMAN-AUTOMATION INTERFACE PROPERTIES

Formal verification has been used to evaluate properties of human-automation interfaces. In the majority of these analyses, the human-automation interface behavior is modeled formally and properties related to the behavior of the interfaces (usually analogous to desirable interface functionality or usability) are checked against this model. A particular subset of human-automation interface analyses are specifically concerned with discovering if there is potential for mode confusion, and there have been several approaches that address this specific problem. However, before formal verification can occur, there must be a formal model. Thus, in this section, we first describe the ways in which human-automation interfaces have been modeled, then discuss general human-automation interface verification, and finally present the different mode confusion analyses.

A. Formal Modeling of Human-Automation Interfaces

The main ways formal models are represented are as finite state transition systems or finite state machines. Human-automation interfaces have been formally modeled as finite state transition systems using different methods [65]. State charts [66] are formal transition systems that support hierarchies, parallelism, and communication that have been used to model interfaces [67], [68]. Interactors are object-oriented interface building blocks that have an internal state and are capable of generating and responding to events [69]. Physiograms are used exclusively for modeling physical device interfaces [70]. Table and matrix specification paradigms like the operational procedure model [71] and ADEPT [72] define interfaces based on their input–output behavior. Abstractions have been developed for formally modeling human interfaces defined in software development environments such as Visual Basic and

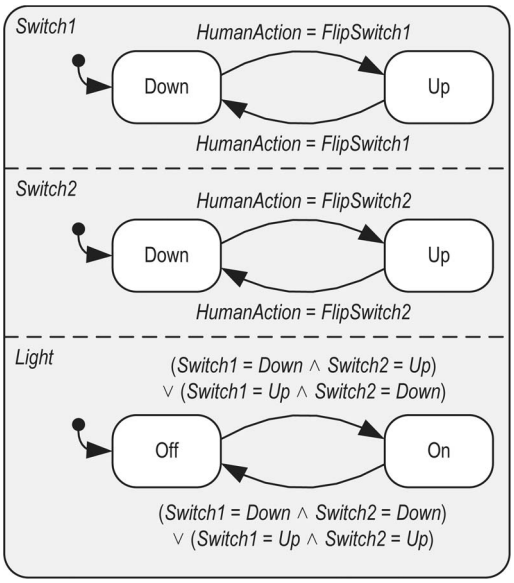

Fig. 1. Illustrative example of a formal (statechart) model of a human-automation interface: a light with two switches. This model is composed of three parallel finite state machines representing two light switches (*switch1* and *switch2)* and the light itself (*Light*). In these representations, a rounded rectangle represents a state, and a labeled arrow between rounded rectangles represents a guarded transition between states. An arrow with a dotted end indicates an initial state. The system receives human action inputs via the $HumanAction$ variable, which can indicate whether the operator has flipped switch 1 or switch 2 ($FlipSwitch1$ and $FlipSwitch2$, respectively). When the human operator flips a switch ($HumanAction = FlipSwitch1$ or $HumanAction = FlipSwitch2$) the corresponding switch will transition between *Up* and *Down*. When only one switch is *Up*, the light is on. Otherwise, the light is off.

Java Swing [73], [74]. Despite the diversity of modeling techniques, all generally treat the interface as having a specific state (a specific valuation of variables) that can transition to different states based on human operator actions, device automation, and/or environmental factors. Some paradigms (such as the interactor) specifically model the rendering of information on the interface display and thus explicitly model what information is available to the operator in any given interface state.

Fig. 1 is a simple example of a formal human-automation interface (a light with two switches) using state charts.

B. Formal Verification of Usability Properties

Work has focused on modeling human-automation interfaces using formal constructs so that their correctness can be evaluated using formal verification. While these analyses do not consider human models as part of the process, they provide guarantees that the human-automation interface will behave in the way it was intended and/or in ways that support safe HAI.

While some research has focused on specifically addressing usability problems with particular classes of human-automation interfaces,[1] there has been a trend toward classifying generic temporal logic property patterns for these purposes. Campos and Harrison [77] identified four related categories of properties that could be expressed in temporal logic and thus formally verified for human-automation interface models: 1) reachability, 2) visibility, 3) task related, and 4) reliability. Reachability properties make assertions about the ability of the interface to eventually reach a particular state. Visibility properties assert that visual feedback will eventually result from an action. Task-related properties describe human behaviors that the interface is expected to support: usually the ability to achieve a particular goal represented by a state or set of states in the interface model. Reliability properties describe desirable interface properties that support safe HAI. Within each of these categories, specific patterns have been identified for checking specific properties (see Table II for some examples).

Thus, with a formal model of a human-automation interface, an analyst can employ the usability property patterns from Table II to formulate specification properties for a given application. Formal verification can then be used to check that the human-automation interface adheres to the properties.

Preliminary work in this area focused on applying these techniques to dialog box-based interfaces. For example, Paternò [80] explored the behavior of different dialog boxes composing a desktop database system. However, these techniques are not limited to dialog box-based interfaces: Campos and Harrison [81] evaluated the air conditioning control panel of a Toyota Corolla; Bolton [82] evaluated the programming interface of a patient-controlled analgesia pump; and Feary [83] explored cockpit interfaces to an autopilot.

An illustrative example can be found in [78] where Campos and Harrison evaluated the interface to a medical syringe pump: a device that delivers medication to a patient via a syringe based on a prescription entered by a human operator. The prescription is programmed into the device on a human-automation interface containing a dynamic LCD screen and several buttons. To assess the general reachability of the interface, Campos and Harrison formulated temporal logic properties to show how to get from each interface state to each other interface state. Thus, for each pairing of 54 interface states, where X and Y represent different interface states, a temporal logic property was generated which took the form in (1). This can be interpreted as "for all possible paths through the model, if the interface is in state X, then there is some future state when the interface will be in state Y.

$$\mathsf{AG}\left((\text{InterfaceState} = X) \Rightarrow \mathsf{EF}(\text{InterfaceState} = Y)\right). \quad (1)$$

When all of these were checked against the formal model of the human-automation interface, they all returned true. Thus, Campos and Harrison were able to prove that the device supported general reachability.

In all of these approaches, any specification violation that is discovered is not necessarily a usability problem; it simply shows that a property suggestive of good usability was not satisfied. As such, any violation needs to be examined by a usability expert and/or explored with usability experiments. The approach's advantage is its ability to discover potential problems previously undiscovered.

These approaches scale when human-automation interfaces (particularly dialog box interfaces) are evaluated in isolation. Abstraction techniques can further ensure scalability. However, there are two barriers to more widespread adoption. It is not standard practice to utilize formal modeling in the design of

[1] See [75], [76] for an example of how formal methods can be used to assess the usability of multimodal, human-automation interfaces.

TABLE II
FORMALLY VERIFIABLE INTERFACE SPECIFICATION PROPERTIES

Category	Property	Informal Description
Reachability	General reachability	It is always possible to eventually reach a specific interface state from a given initial interface state [78].
	State inevitability	It will always be true that a specific interface state will eventually be reached [79].
	Weak reachability	A specific action will always result in a change in the interface state [80].
	Strong reachability	A specific action will allow for the possibility of a future change in interface state [80].
Visibility	Feedback	A specific action will always result in a change in interface state that is available to the human operator [80], [81].
	Continuous feedback	All human operator actions must produce a change in interface state that is available to the human operator and must do so before any additional actions can be performed [80].
Task related	Weak task completeness	There is at least one action sequence from a specific initial interface state that will eventually achieve a specific goal [79], [80].
	Strong task completeness	For any given possible sequences of actions from a specific initial interface state, there is a set of additional actions which will eventually achieve a specific goal [79].
	Weak task connectedness	From any interface state, there is at least one action sequence that will achieve a specific goal [79].
	Strong task connectedness	From any interface state, there is at least one action sequence that will eventually achieve a specific goal with a specific final action [79].
	Undo / Reversibility	The effects of a specific action can be undone with a single additional action or eventually undone with at least one sequence of actions [79]–[81].
Reliability	Behavioral consistency	A specific action will always result in a change in interface state that adheres to a specific characterization [81].
	Rule set connectedness	There is at least one situation in which the interface supports the ability to perform a specific action through the interface (such as clicking a button) [79].
	Deadlock freedom	The interface will never reach a state that will never accept human operator input [79].
	State floatability	The human operator can go from one specific interface state to another without ever reaching an undesirable state [79].

human-automation interfaces. Also, temporal logic properties can be difficult to formulate correctly, and (as was true in the example above) a large number may be required.

To address the first problem, researchers have constructed design and analysis environments that allow formal models to be developed more easily. Degani *et al.* [84] has shown how design patterns can be used when constructing formal human-automation interface models with state charts; Campos and Harrison [78] have developed the IVY tool that allows human-automation interface models (constructed from interactors) to be graphically prototyped; Dwyer *et al.* have created tools that allow human-automation interfaces defined in Visual Basic and Java Swing to be automatically translated into formal models [73], [74]; and ADEPT [72], [83] allows formal human-automation interface models to be created using a matrix/table interface that does not require knowledge of formal modeling or computer programming.

The second problem is also being addressed. In addition to patterns for specifying all of the different usability properties from Table II [79]–[81], tool assistance has also been explored. For example, IVY [78] and its precursor IFADIS [85] both allow analysts to use usability temporal logic patterns for reachability, visibility, and reliability properties to be easily applied to models constructed in their environments. ADEPT [83] offers the ability to evaluate visibility and reliability properties automatically. Bolton [82] introduced a method to automatically generate task-related properties from task analytic behavior models.

C. Formal Verification and Mode Confusion

A specific extension of the work on the formal verification of human-automation interfaces has focused on identifying conditions in interfaces that could lead to mode confusion. Some of these efforts are similar to the usability work discussed above in that they focus on formally verifying properties against a formal model of the human-automation interface. However, other efforts have included some rudimentary models of the human operator as a means of gaining insights on mode confusion.

1) Analyses Using Human-Automation Interface Models: The first set of analyses focused on finding potential mode confusion with only a model of the human-automation interface and the basic underlying automation. Leveson *et al.* [86] identified a number of human-automation interface properties that were associated with mode confusion (Table III).

Researchers [87]–[91] have explored how formal verification processes with both model checkers and theorem provers can find these types of conditions in formal models of human-automation interfaces for medical devices, aircraft flight guidance systems, and mode control panels. In the majority of these evaluations, the analyses explore how transitions occur within a model's state space (for example, finding unexpected ways of reaching off-normal states), attempting to discover if there are situations in which human operator expectations about how their actions impact the state of the interface (encoded in temporal logic) are satisfied by the interface (for finding unintended side effects, indirect mode changes, operator authority limits, and inconsistent behavior); determining when two interface states display the same information (for finding conditions with a lack of appropriate feedback); and determining if there are interface states that do not properly represent the state of the automation (also for human-automation interface states lacking appropriate feedback).

For a simplified model of the modes in a flight guidance system, Joshi *et al.* used both a model checker (NuSMV [92]) and a theorem prover (PVS [93]) to show how different mode confusion problems could be discovered. Despite the example's

TABLE III
SYSTEM AND INTERFACE PROPERTIES ASSOCIATED WITH MODE CONFUSION

Name	Description
Off-normal transitions	A system transitions into an "off-normal" state (one not associated with normal system operation).
Unintended side effects	A action does more than what was intended by the human operator.
Indirect mode changes	A system changes state without human-operator input.
Operator authority limits	There are limitations on operator authority which result in operator actions being ignored in some contexts.
Inconsistent behaviors	The same human action achieves different effects in different contexts.
Lack of appropriate feedback	The human-automation interface does not provide enough information for the human operator to determine what mode the system is in.

simplicity, they were able to show how each entry in Table III could be identified.

Using the IVY environment, Harrison [91] modeled an infusion pump that automatically delivers medicine to a patient intravenously. Different specification properties were formulated to check for the mode confusion properties in Table III. One of the most compelling addressed the lack of appropriate feedback. Specifically, Campos and Harrison identified three messages displayed on the top line of the device's LCD screen (coded here as Messag1, Message2, and Message3) that indicated medication delivery. They then ran a check to ensure that these were necessary and sufficient for the pump to be delivering medication, represented in temporal logic as in (2).

$$\mathsf{AG}\left(\begin{array}{c}(\text{InfusionStatus} = \text{Infuse}) \\ \Leftrightarrow \left(\begin{array}{c}\text{TopLine} = \text{Message1} \\ \vee\text{TopLine} = \text{Message2} \\ \vee\text{TopLine} = \text{Message3}\end{array}\right)\end{array}\right). \tag{2}$$

When this was checked against the formal model, it failed. The counterexample revealed a specific scenario in which a function key on the device could be pressed that would replace the top line of the LCD with a different message while medication was being delivered. Thus, there was a state in which the device's display would not indicate that the pump was infusing, allowing for potential mode confusion.

This type of formal verification can provide insights into conditions that might result in mode confusion. However, the models did not include any information regarding the human operator's knowledge about current and expected future system states and/or future modes and modes changes. Therefore, it is not clear that mode confusion will occur even if a violation is found. To address this problem, two separate approaches allow potential mode confusion to be explicitly discovered using models of the human operator. Both are explored next.

2) Analyses Using Human Mental Models: In the first approach allowing for the explicit detection of mode confusion, a human operator mental model of how the automation works (presumably an abstraction) is formally modeled along with the device or system automation.[2] The resulting composite model is then checked (either manually or using model checking) to find inconsistencies between the human-device mental model and the automation: when the human operator's mental model state does not match or is not the proper abstraction of the automation model's state. Degani, Heymann, Oishi *et al.* [67], [94]–[96] showed how inconsistencies could be algorithmically discovered between state chart representations of the human operator's mental model and device automation in a variety of applications including an aircraft autopilot [67], a cruise control system [94], and an aircraft auto-land system [95]. Sherry *et al.* [97] used matrix-based representations of both the device automation's input–output behavior and the human operator's mental model of that input–output behavior and showed how they could be algorithmically compared to find potential mode confusion for an aircraft's mode control panel. Rushby *et al.* [98], [99] used Murϕ[3][100] and Buth [101] used FDR2 [102] to show how model checkers could be used to find mode problems in the MD-88 autopilot.

These examples highlight situations where pilots may miss mode transitions and misunderstand how particular modes or mode transitions work. Most highly automated aircraft include multiple pitch modes. The automation will automatically transition between these modes when it is trying to change and then hold the aircraft's altitude. A pitch hold mode, for example, can maintain a particular pitch attitude until the aircraft climbs (or descends) to the higher (or lower) commanded altitude. Vertical speed mode can maintain a particular vertical speed until the commanded altitude is reached. A flight level change mode can maintain a particular airspeed until the commanded altitude is reached. In both cases, if an altitude capture mode is armed, an automatic transition to an altitude hold mode should occur to maintain the commanded altitude once the desired altitude is achieved. The interaction of the pitch modes and the arming of altitude capture mode have been linked to mode confusion and the violation of associated altitude clearances [24]. Such scenarios are particularly dangerous because they can allow an aircraft to climb into another aircraft's airspace. By constructing formal models representing an abstraction of a pilot's understanding of pitch modes and actual pitch mode operations, researchers have been able to replicate conditions that can lead to violations of altitude clearances [98], [99], [101].

Because this approach requires the modeling of both the device automation and the human operator's mental model, it is less scalable than the previous one only requiring a model of the human-automation interface. It is not surprising that

[2]In these analyses, the human-automation interface model may be implicit in the interaction between the mental model and the automation model (the automation receives human actions and the mental model receives display information from the automation) or modeled explicitly as a finite state transition system that interacts with the mental model and the automation model.

[3]Pronounced MUR-fee.

the evaluated models have been fairly simplistic and/or abstract. Given the discrete nature of formal models, many of the systems that have been modeled are discrete in nature or have been abstracted to be such. However, the use of hybrid systems techniques have been explored to allow the dynamics of aircraft to be included in these analyses [95], [96], [103]. Most recently, Rushby *et al.* [103] showed how emerging abstraction techniques can be used with an infinite bounded model checker [104] to evaluate system models containing continuous quantities. The authors demonstrated how a known problem with an older version of the "speed protection" of an Airbus A320 could be discovered with this method.

An extension of this approach has focused on ways of generating human-automation interfaces that do not exhibit properties associated with mode confusion. These methods assume that the human operators must have the information necessary to maintain a correct mental model through the human-automation interface. Thus, a mental model representing an abstraction of the system's device automation that does not facilitate mode confusion can be interpreted as a specification for the information content and behavior of the human-automation interface. Crow *et al.* [105] discussed how mental models constructed through results from questionnaires can be progressively abstracted and/or refined and checked with a model checker until a "safe minimal mental model" is found: the smallest mental model that does not facilitate mode confusion. Heymann and Degani [106] proposed an algorithmic means of achieving such a model. First, a state chart representation of the system's automation is constructed. Then, each state is assigned to one of several specification classes: aggregate states that the analyst hypothesizes the human operator must distinguish between. This model is then systematically reduced by combining states within specification classes in a way that avoids inconsistencies indicative of mode confusion. Combéfis and Pecheur [107] showed how such a process could be performed automatically without the need for specification classes. This approach has been further extended to allow the machine learning algorithm to automatically learn a minimal human-automation interface model from a device automation model [108].

A limitation of this approach is that it assumes the human operator creates a mental model of how the system's automation behaves, and this model can be represented as a formal finite state transition system. Extracting such a model from a human operator may be difficult, though several approaches have been explored [105], [109]. Further, Norman [110] notes that actual human operator mental models are incomplete, are difficult for the human operator to "run," are unstable, do not have firm boundaries, are unscientific, and are "parsimonious" (overly abstract to save on mental complexity).

3) Analyses Using Human Knowledge Models: Rather than attempt to capture the operator's mental model of how the system works, a third approach uses formal models representing the human operator's knowledge about how he or she accomplishes goals with the system through the human-automation interface [111]–[113] to find potential mode confusion. Effectively, human operators have knowledge about how they accomplish goals with the system based on their experience and their perceptual observations when interacting with the system. These can be represented formally as processes (for example see [114]). When paired with human-automation interface and device automation models, a model checker can determine if there are states in the system that block the human operator process from accomplishing his or her goals or if there are ways of achieving goals that do not match the procedures in the operator's knowledge. Both can result in mode confusion. Wheeler [113] illustrated how such a method could be performed manually to evaluate the alarm states of a digital alarm clock. Bredereke and Lankenau [111], [112] and Bredereke [115] automated this type of evaluation using communicating sequential processes (CSP) [114] and the FDR2 model checker [102]. Doing this, they were able to find a number of potential mode confusion problems with the control system of an electric wheel chair and various features of a telephone communication system.

With a telephone system example, Bredereke [115] found a number of potential mode confusions. One pertained to the call waiting system. When someone is using a phone with call waiting, the person knows that he or she can accomplish the goal of putting callers on hold. However, phones may have a feature that prevents emergency numbers (such as 911 in the United States) from being put on hold. This can result in a potentially dangerous mode confusion. Assume Person A calls and talks to Person B. During the course of this conversation, A decides that he or she needs to consult emergency help because of a situation with B. A puts B on hold and calls 911 for emergency help and is connected to Person C. A then wishes to relay emergency information from C to B by putting C on hold and talking to B. However, because C was reached via an emergency number, Ccannot be put on hold and A cannot relay the safety critical information back to B. The fact that A is blocked from putting C on hold indicates a mode confusion.

To be useful, this approach requires that analysts capture the potentially imperfect knowledge the human operator has about achieving goals with the system. Bredereke and Lankenau [112] discuss several ways that this can be accomplished. Although not currently explored, given that the knowledge being encoded is procedural, such a method could exploit task analyses [37], [38] for model development.

Javaux [116] investigated an automated means of constructing realistic, imperfect knowledge models. Javaux starts with a model that contains all of the knowledge the operator needs to interact with the system. Then, by evaluating how often human operators have to use specific knowledge, he allows the rarely used section of the knowledge model to degrade based on the theory of implicit learning. Javaux showed how this can lead to knowledge models that facilitate mode confusion in an autopilot system under rare operational conditions.

There are no studies that directly compare this approach to the mental model approach discussed above. Thus, while both explicitly model mode confusion in the human operator, it is not clear which is more adept at discovering problems nor is it clear which analysis will scale better.

III. Formal Verification of System Properties

The analyses presented in the previous section are convenient in that HAIs human-automation interfaces are often easily represented as formal models. Further, the analyses are inherently limited in scope, only being concerned with the

human-automation interface and thus avoiding the increases in model size potentially associated with modeling the full system. However, because human behavior is not explicitly modeled in any of these techniques, they can only be used to find system conditions theorized to be preconditions to system failures. Another category of approaches have focused on identifying how modeled human task analytic behavior can potentially contribute to system problems beyond potential shortcomings with the human-automation interface. When designing the procedures, displays, controls, and training associated with the HAI of an automated system, human factors engineers use task analytic methods to describe the normative human behaviors required to control the system [38]. The resulting task analytic models represent the mental and physical activities operators use to achieve the goals the system was designed to support. There are a number of different task analytic methods, and thus many different forms that task analytic models can take. However, a distinction can be made between models that are primarily concerned with capturing the observable manifestation of human behavior (henceforth referred to as task models), and those that are concerned with describing the cognitive process that drives the observable human behavior (henceforth referred to as cognitive models). Both types of models have been used in formal verification analyses.

A. Formal Verification and Task Models

Task analytic models representing the observable manifestation of human behavior are typically structured as a hierarchy, where activities decompose into other activities and (at the lowest level) atomic actions. Task models such as concurtasktrees (CTT) [117], operator function model (OFM) [118], enhanced OFM (EOFM) [119], user action notation (UAN) [120], or AMBOSS [121] can be represented using discrete graph structures as hierarchies of goal-directed activities that ultimately decompose into atomic actions. In these models, strategic knowledge (condition logic) controls when activities can execute and modifiers between activities or actions control how they execute in relation to each other. Because they can be represented discretely, task models can be used to include human behavior in formal system models along with other system elements including device automation, human-automation interfaces, and the operational environment [4], [122].

Some researchers have modeled human tasks in the employed analysis package's formal notation. Degani *et al.* [123] incorporated human task models into state chart models of human-automation interfaces and used them to explore human operator behavior during an irregular engine-start on an aircraft. Basnyat *et al.* [124], [125] used a Petri net-based formalism called interactive cooperative objects (ICO) to model human task behavior as part of larger system models (a waste fuel delivery plant and a satellite control system) to verify properties critical to the system's safe operation. Resolutions to discovered problems came in the form of barrier systems to monitor the system and prevent it from transitioning to unsafe states [124] or through modification to operator training materials [125]. Gunter *et al.* [126] encoded patterns of human task behavior into CSP concurrently with system and environment models (also encoded in CSP) to verify safety properties of a portable automated identification and data capture device used to identify and record data about patients and hospital equipment. They showed that a "protection envelope" could be incorporated into the system's automation to ensure that the modeled normative human task behavior would never result in situations where incorrect or corrupted data could be entered into the device.

These approaches require practitioners to encode tasks in formal modeling languages. Thus, some researchers have used more established task analytic modeling notations. These are then translated into the needed formalism. Palanque *et al.* [127] showed how task models written in UAN [120] could be translated into ICO and used to verify task behaviors for interacting with an automated teller machine (ATM). Fields [128] developed a custom notation for modeling hierarchies of activities in human tasks along with the temporal relationships between them. Aït-Ameur *et al.* [129], [130] and Paternò and Santoro [131] have translated CTT [117] into formal models and performed formal verification with larger system models. Aït-Ameur's work used the Event B theorem prover to verify human interaction with a software dialog box for converting currencies. Paternò *et al.* translated CTT models of multiple human operators managing runway traffic into LOTOS [132] and then into a formal model where it could be checked as part of a system model encompassing the behavior of the interface, its automation, and the environment. Bolton and Bass *et al.* [119] have developed EOFM (an extension of OFM [118]) as a generic, XML-based task modeling language. They have also developed a translator that converts EOFM models into code for the Symbolic Analysis Laboratory [133] for use in the formal verification of safety properties of larger system models. They have applied their technique to a patient controlled analgesia pump [122], [134], an automobile with a cruise control [119], and a pilot performing the before landing checklist [4].

Bolton *et al.* [119] presented an example with an automobile with a simple cruise control system. The cruise control system had two buttons for enabling and disabling cruise. This system was modeled as part of a general automobile driving interface involving the accelerator and the brake pedals. The automobile was modeled as moving down a highway with a traffic light at the end. Midway along the highway was an area in which traffic could merge. The driver could drive at a specific, desired speed. A task model was created in EOFM. The driver could 1) drive at the desired speed (by manipulating the accelerator and brake pedals and/or the cruise control), 2) avoid merging traffic (by speeding up or slowing down), and 3) respond to the the traffic light based on its relative distance and color. An analysis checked whether there could be a situation in which the driver could run the traffic light. In temporal logic, this can be encoded as shown in (3), interpreted as there should never be a condition where the car is at the intersection when the light is red and the car is not stopped

$$\mathsf{AG}\neg \begin{pmatrix} \text{TrafficLightDistance} = \text{AtIntersection} \\ \wedge \text{TrafficLight} = \text{Red} \\ \wedge \text{Car} \neq \text{Stopped} \end{pmatrix}. \quad (3)$$

When checked with the complete system model, the specification property could be violated. An interpretation of the counterexample [135] revealed that this occurred if: 1) the driver reached his or her desired speed and enabled the cruise control; 2) when the merging car arrived, the driver accelerated to go ahead of it and stayed at that higher speed; 3) the driver removed his or her foot from the gas pedal to respond to a red traffic light; and 4) once the car slowed to the cruise speed, it stayed at the cruise speed, causing the car to not stop at the intersection.

While the majority of these types of analyses have focused on systems that use a single human operator, progress has been made on extending these techniques to multi-operator systems. Paternò *et al.* [136] and colleagues identified three different ways that collaborative human behavior could be incorporated into formal task models: 1) A single task model could be created describing the behavior of every human operator in the system; 2) Separate task models could be created for each human operator with constructs being used to describe the coordinated and communicative relationships between activities and actions between operator tasks; and 3) A separate task model could be created for each human operator but with the addition of a shared structure which relates the coordinated and communicative relationships between the activities and actions contained in the separate models. Bass *et al.* [137] presented a fourth means of accomplishing this: allowing separate tasks to be created for each operator for activities not requiring coordination, and allowing for separate shared tasks/activities that require multi-operator coordination and communication.

These analyses are more limited by scalability than the human-automation interfaces analyses. Since they are more concerned with system failures, they require larger models than those that are only concerned with human-automation interfaces. Results from Bolton and Bass [122] indicate that analyses only using task behavior models can reduce the overall size of model state spaces when they are paired with large system models. However, this may not be true for all models, particularly those with multiple human operators.

Another potential limitation of this approach is that it does not consider erroneous human behavior, which is often associated with failures of human-automation interactive systems. However, the community had taken steps to address this. Fields [128], Bastide and Basnyat [138], Bolton *et al.* [139], Bolton and Bass [140], Paternò and Santoro [141] have investigated how patterns of erroneous behavior (based on the phenotypes of erroneous actions [142]) can be manually incorporated into task analytic behavior models. These models could then be integrated with the analyses described above to use formal verification to investigate whether these errors impact the fulfillment of specification properties. Bolton *et al.* [143] and Bolton and Bass [144] have extended this work so that erroneous human behavior can be automatically generated in task models, allowing analysts to determine how potentially erroneous human behavior can contribute to system failures without having to anticipate which behaviors might be problematic. One method [143] creates a task structure that generates zero-order phenotypes of erroneous actions [142] for each action in the task model, but limits the total number of possible erroneous acts to an analyst specified maximum. The second method [144] manipulates the interpretation of strategic knowledge in the task model to replicate slips of attention [12], again limiting the total number of erroneous behaviors with a maximum. These methods were used to discover problems related to controlling a medical radiation therapy machine [143] and a patient controlled analgesia pump [144]. Unfortunately, such analyses have the potential to reduce scalability, limiting the maximum number of erroneous human behaviors that could be considered in any given analysis. Such techniques also have the disadvantage that they do not describe a cognitive mechanism for any of the modeled erroneous behaviors.

B. Formal Verification and Cognitive Models

Different analyses have attempted to model task analytic human behavior using high fidelity, albeit abstract, cognitive architectures in formal verification. The goal has been to model the cognitive process the operator employs to decide what actions he or she will use to interact with the system. These methods let the analyst to formally verify that the system will always allow the operator to achieve his or her goals with a set of cognitive behaviors. These methods can identify situations where the human operator fails to achieve desired goals or drives the system into dangerous operating conditions.

These cognitively based models of human behavior focus on modeling the knowledge human operators use for interacting with a system. The human operator's cognitive state (a set of variables) changes in response to the state of the human-automation interface, the operation environment, or the operator's current cognitive state according to a set of logically defined production rules. Ultimately, this can result in rules allowing the human operator to perform actions (themselves represented by changes in variable values). The nature of these models and the analyses they support are determined by the cognitive architecture in which they are instantiated.

Lindsay, Connelly *et al.* [145], [146] used a cognitive architecture called the operator choice model (OCM) specifically created for an an en-route, air traffic control task, in which a human operator monitors an airspace and attempts to resolve potential conflicts between aircraft. The OCM describes the process human operators use to scan or search human-automation interfaces for information (scan), determine if that information is worthy of additional interest (classification), decide how to proceed in light of the assessed information (decision making), and execute a plan of actions (perform action). The cognitive model was specified as a set of event-driven transitions (rules) for each of these four processes, where control flows between each of these processes in the cognitive model. The human-automation interface and the operational environment (the state of the airspace) were modeled as separate processes. Temporal logic patterns were created for potential failures in the air traffic control task: not resolving an actual conflict, resolving a non-existent conflict, and creating a conflict where one previously did not exist. Temporal logic properties specified that problems in a specific cognitive process (scanning, classification, decision making, or action performance) could result in one of the three failures. A model checker was used to determine if and in what context any of these failures could occur.

While useful, the OCM model is limited in that it is only relevant for air traffic control tasks and thus not generalizable. However, Blandford, Curzon, *et al.* [147]–[150] have focused

on creating models around a more generic cognitive architecture from programmable user models (PUMs) [151]. In these models, the human operator has a set of goals he or she may want to achieve with the system, and a set of actions to interact with it. A set of rules (referred to as beliefs or knowledge) define when a human operator may attempt to pursue a specific goal based on the state of the human-automation interface, the operational environment, or other goals being pursued. Actions can ultimately be performed when a human operator commits to applying one according to a specific rule. Note that in these models, the human operator commits/decides to perform an action before that action is actually executed. Such models have been used with human initiated systems (where system behavior is solely driven by its human operator) [148] and evaluated using formal verification with both theorem provers [149] and model checkers [152].

These formal PUM-based analyses have been used to obtain different insights about systems. Butterworth *et al.* [149] showed how specification properties related to system safety, liveness, and usability could be investigated for a simple web browsing task. Butterworth *et al.* [149] used PUMs and formal verification with the same application to identify the type of knowledge a human operator requires to successfully fulfill his or her goals. These analyses have been extended to identify cognitively plausible erroneous behaviors based on a human operator model interacting with an automated system. These include repetition of actions, omission of actions, committing actions too early, replacing one action with another, performing one or more actions out of order, and performing an unrelated action [34]. Means of identifying post-completion errors (special types of omission errors in which the operator forgets to perform actions that occur after the completion of a high-level goal [33]) have also been identified and illustrated using a model of a vending machine [35]. Design rules were applied to address these errors, and their effectiveness was evaluated using the same formal verification process [34], [35]. Work has also investigated how to use these types of models to determine when different types of operators (expert versus novice) may commit errors when interacting with an ATM [153]. The architecture has been extended to a generic cognitive architecture that models the effects of salience, cognitive load, and the interpretation of spatial cues. It can assess whether they result in erroneous human behavior when interacting with an ATM machine [152], [154], [155]. Basuki *et al.* [156] used heuristics for modeling human operator habituation, impatience, and carefulness within such a cognitive architecture and illustrated erroneous human behaviors for interacting with the vending machine model from Curzon and Blandford [35].

Consider the ATM example from [153]. Curzon *et al.* formally modeled cognitive rules for selecting primary goals and secondary goals, selecting actions to achieve these goals, performing actions, reactive behavior based on information conveyed by the human-automation interface, and terminating goals. For example, when interacting with the machine, the human operator may have the primary goal of extracting cash from the machine. However, he or she may also have secondary goals related to obtaining a transaction receipt and having the ATM card returned. There are a number of actions the human operator needs to select and execute to perform his or her goals. These include inserting the ATM card, entering the pin, selecting an appropriate amount of cash, and so on. The model has rules for defining how the operator can react to prompts and alarms. The cognitive model can also have rules defining when to terminate behavior once goals have been achieved. The cognitive model was paired with a formal model of the ATM. The entire system was evaluated with a theorem prover [93] to uncover potential situations where the human operator finishes interacting with the machine without completing all of his or her goals. However, this verification procedure revealed a post-completion error scenario in which it was possible for the human operator to receive his or her cash and terminate interaction with the machine (because the primary goal was achieved) while leaving the ATM card in the machine (not achieving a secondary goal). Curzon *et al.* then introduce an improved machine design that requires the human operator to retrieve the card before cash is dispensed. They then use the theorem prover to show that the new design does not have the previously discovered problem.

Masci *et al.* [157] have introduced a method for modeling and evaluating distributed cognition in the PVS theorem prover [93]. The models used by Masci *et al.* were primarily focused on information flows between entities and the associated cognitive processes for a simple model of the London Ambulance Service. Given the nature of the approach, cognitive models of individuals were kept very simple. Current effort is extending these cognitive modeling approaches to multi-operator systems. Future work will determine how well these cognitive modeling approaches will scale.

All of the cognitive modeling approaches have a distinct advantage over those that model only task behavior as analysts can infer the cognitive reason for erroneous behavior. Additionally, erroneous human behavior can arise organically from the cognitive model rather than having to be specifically inserted or generated in task analytic behavior models. However, cognitive models are more difficult to construct than task analytic models. To date, there have been no studies comparing the cognitive modeling approaches to the task analytic behavior modeling approaches, so it is not clear how they scale relative to each other. Obviously, cognitive models have the potential to be more complex, thus less likely to scale, than task models. However, given the more naturalistic way that erroneous behavior is modeled in the cognitive approaches, it is possible that cognitive model approaches will scale better than those based on task models when erroneous behavior is the concern. Future work should attempt to compare these approaches so that the tradeoffs between them can be better understood.

IV. Discussion and Future Development

Formal verification offers possibilities for evaluating HAI not offered by other analysis techniques (such as simulation and testing) through its exhaustive means of proving or disproving the validity of desirable properties. Thus it provides additional ways of finding potential shortcomings in human-automation interfaces and/or of discovering and diagnosing system failures related to human operators.

The specific evaluation approaches discussed herein cover the techniques the HAI community has successfully employed despite the limitations of current formal verification technology. Advances in formal methods are forthcoming and will support new types of HAI analyses with formal verification. In this section we discuss how advances in formal methods may ameliorate some of the current limitations of formal verification. We also discuss how formal verification can be integrated into other types of HAI analyses.

A. Advances in Formal Methods

1) Tradeoffs Between Techniques and Analysis Integration: There are tradeoffs between the presented evaluation techniques. The analyses of human-automation interfaces for usability and mode confusion scale better than the larger system analyses that use task and cognitive models. However, the human-automation interface analyses only look for conditions that could result in system problems rather than system problems themselves. Conversely, the system-level analyses that utilize task models or cognitive models as part of larger formal systems models are capable of discovering such problems.

The current state of the technology is such that many of the presented techniques are not integrated. For example, an analyst attempting to perform a system analysis using task models will likely need to alter or re-implement the system model to evaluate the human-automation interface. This is largely due to the different components that are modeled in each technique. Subtle differences between model elements shared by the different techniques (such as the human-automation interface) can exist because of the different interactions that may occur between modeled elements. Some work has partially addressed this issue. Environments such as IVY [78] allow for a number of usability analyses to be performed. However, these do not currently support the modeling of the human operator. Bolton and Bass [122] have utilized an architectural framework that separates system model elements to allow for more flexible analyses. This has primarily been used in system evaluations using task models [4], [119], [143], [144], but has also been used to evaluation of human-automation interface properties [82]. Rukšėnas and Curzon [158] have shown that their cognitive modeling system can be used to find mode confusion problems in addition to being used for system analyses. However, what is needed is an integrated modeling environment that will allow analysts to easily perform multiple analyses with minimal changes between components. Such an environment would allow analysts to take full advantage of the formal HAI verification analyses that are currently available.

2) Model Validity and Design Integration: In the formal methods community, languages like LUSTRE [159] and the associated Safety Critical Application Development Environment [160] allow systems to be "correct by construction": designers specify models of embedded, reactive, synchronous systems; formally verify them; and generate C and Ada code guaranteed to exhibit the verified properties. This allows formal verification to be integrated into the design of the end product, thus avoiding problems of validity that can arise when formal models are independent from designs.

There have been attempts to integrate the design of human-automation interfaces into such integrated development environments for the purpose of facilitating some of the analyses discussed here. Several groups have adapted their modeling frameworks to work with human-automation interfaces built in Java and Visual Basic, thus allowing the actual software implementations to be used in the formal analyses of human-automation interfaces and system properties with human-task behavior models [74], [78], [130]. Some have also shown that it is possible to execute formal models as if they were computer programs while allowing human operators to interact with them [78], [130], [161]. They have used this ability to experimentally validate human task behavior models against actual human behavior, ensuring that the modeled human behavior conforms to the behavior exhibited by actual human operators in human subject experiments. Similarly, such environments allow human operators to interact with the human-automation interfaces rendered in formal design environments [72], [78], [83], [130], [161]. This can allow usability studies to be conducted with human subjects. Masci *et al.* [157] have used simulations of their formal distributed cognition models to allow subject matter experts to judge model validity. Huang *et al.* [162] have shown how formal cognitive models can be experimentally validated. Work on these types of integrated development environments is in the early stages. Future work could result in tools that transcend the boundaries between modeling, design, and implementation.

3) Model Scalability: As formal modeling and verification technology improves, the size of the systems they can accommodate will increase. Additionally, future developments in lightweight formal methods [163], [164] may prove useful as they allow parts of a system to be formally modeled and verified using compact modeling languages that can be quickly verified. Formal architectural frameworks like the one used by Bolton and Bass [122] (where a system is composed of separate formal models of the human mission, human task behavior, human-automation interface, device automation, and operational environment) may be useful in such lightweight analyses as they could allow certain system submodels to be eliminated for certain analyses. Light weight modeling techniques like those offered by ADEPT [72], [83] allow human operators to prototype human-automation interface behavior and quckly check a number of properties that might suggest problems with HAI. Work related to formally modeling and reasoning about the role of human behavior's contribution to system failures in accident reports [125], [165]–[168] may also suggest lightweight or compositional analyses that only consider the safety critical elements of systems. Emerging abstraction techniques [103] will also help with scalability.

B. Supporting Other HAI Analyses With Formal Verification

Even with advances in formal methods, the state explosion problem will most likely remain a limiting factor for model checking analyses. Additionally, there has been little to no work investigating how common HAI metrics like timing, workload [169], [170], situation awareness [46], [171], trust [172], and compromise [50] can be extracted from verification analyses.

Thus, the use of formal verification in the evaluation of human-automation interactive systems may be more successful at complementing other forms of HAI analyses. This would allow the exhaustive nature of formal verification to complement the advantages offered by simulation, experimentation, and human performance modeling.

1) Formal Verification and Simulation: While formal verifications are exhaustive, they do not scale well. On the other hand, simulations are rarely exhaustive, but are much more scalable. As such, some degree of success has been found in using formal verification to selectively evaluate bounded elements of a system within a simulation framework [173], [174]. While these techniques have traditionally been used to evaluate computer hardware, they could be adapted for use in the evaluation of human-automation interactive systems. In such a framework, simulations could be used evaluate large systems and formal methods would be periodically applied to the evaluation of human interaction with human-automation interface. In fact, some work is already heading in this direction [137], though more work is needed.

2) Formal Verification and Human Subject Experimentation: Human subject testing has high face validity where actual people interact with the real system. It also supports collection of many human performance metrics. However, because conditions that lead to system failure may occur under rare or unexpected conditions, human subject testing is a time consuming, expensive, and inefficient way of finding human-automation interactive problems that occur in these rare events. To this end, formal verification could be used to influence the design of human subject testing. By using formal task analytic models of the human operator paired with an abstract model of the actual system, formal verification could be used to exhaustively search through the system to determine if there are conditions where HAI could be potentially problematic. Analysts could then use human subject testing to explore whether the potentially identified problems can actually occur and explore how to correct them. Such a method offers advantages beyond using either of these two techniques separately. The formal verification allows for the entire system's statespace to be exhaustively examined, which cannot be done with human subjects. However, because analyses will ultimately be investigated with human subjects, the analyses will have higher face validity and system models can be rendered more abstractly than they would be otherwise, mitigating problems related to scalability.

3) Formal Verification and Human Performance Modeling: Many of the analyses discussed here are of similar scope to those of human performance modeling. Infrastructures such as CogTool [175] allow analysts to specify human-automation interfaces and human-task behavior using a simple point-and-click interface. These are then coupled with a sophisticated ACT-R [176] cognitive model that allows the analyst to predict how long the human operator will take to perform tasks with the system. Given the formality with which these are specified, it is likely they could be incorporated into the formal verification analyses similar to those discussed here. For example, Rukšėnas *et al.* [177] have investigated ways of performing keystroke-level timing analysis (similar to that used by KLM-GOMS [52]) with human behaviors generated using their cognitive-modeling and formal verification infrastructure. Potential exists for further integrating human performance modeling and formal verification analyses where models of trust, situation awareness, and workload could be integrated into formal models or, as with the timing analysis of Rukšėnas *et al.*, the output of formal verifications could be made to work with such models. Future work should attempt to bridge the gap between these two areas.

REFERENCES

[1] BASI. (1998). Advanced technology aircraft safety survey report, Dept. Transp. Regional Develop., BASI, Civic Square, Canberra, Australia, Tech. Rep. [Online]. Available: http://www.atsb.gov.au/media/704656/advancedtechnologyaircraftsafetysurveyreport.pdf

[2] FAA Human Factors Team. (1996). Federal aviation administration human factors team report on: The interfaces between flightcrews and modern flight deck systems, Federal Aviation Admin., Washington, DC, Tech. Rep. [Online]. Available: http://www.faa.gov/aircraft/air_cert/design_approvals/csta/publications/media/fltcrews_fltdeck.pdf

[3] D. Hughes and M. Dornheim, "Accidents direct focus on cockpit automation," *Aviation Week Space Technol.*, vol. 142, no. 5, pp. 52–54, Jan. 1995.

[4] M. L. Bolton and E. J. Bass, "Using model checking to explore checklist-guided pilot behavior," *Int. J. Aviation Psychol.*, vol. 22, no. 4, 2012, to be published.

[5] J. M. O'Hara, J. C. Higgins, W. S. Brown, R. Fink, J. Persensky, P. Lewis, J. Kramer, A. Szabo, and M. A. Boggi, "Human factors considerations with respect to emerging technology in nuclear power plants," U.S. Nucl. Regul. Comm., Washington, DC, Tech. Rep. NUREG/CR-6947, 2008.

[6] L. T. Kohn, J. Corrigan, and M. S. Donaldson, *To Err Is Human: Building a Safer Health System*. Washington, DC: Nat. Acad. Press, 2000.

[7] P. Ladkin, AA965 Cali accident report: Near Buga, Colombia, Dec. 20, 1995, Bielefeld Univ., Bielefeld, Germany, 1996. [Online]. Available: http://sunnyday.mit.edu/accidents/calirep.html

[8] E. Sekigawa and M. Mecham, "Pilots, A300 systems cited in Nagoya crash," *Aviation Week Space Technol.*, vol. 145, no. 5, pp. 36–37, Jul. 1996.

[9] "Grounding of the Panamanian passenger ship Royal Majesty on Rose and Crown Shoal near Nantucket, Massachusetts June 10, 1995," Nat. Transp. Safety Board, Washington, DC, Tech. Rep. MAR-97/01, 1997.

[10] C. Perrow, *Normal Accidents: Living With High-Risk Technologies*. Princeton, NJ: Princeton Univ. Press, 1999.

[11] N. G. Leveson and C. S. Turner, "An investigation of the therac-25 accidents," *Computer*, vol. 26, no. 7, pp. 18–41, Jul. 1993.

[12] J. Reason, *Human Error*. New York: Cambridge Univ. Press, 1990.

[13] T. B. Sheridan and R. Parasuraman, "Human-automation interaction," *Rev. Human Factors Ergonom.*, vol. 1, no. 1, pp. 89–129, Jun. 2005.

[14] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 30, no. 3, pp. 286–297, May 2000.

[15] L. Bainbridge, "Ironies of automation," *Automatica*, vol. 19, no. 6, pp. 775–779, Nov. 1983.

[16] E. Roth, K. Bennett, and D. Woods, "Human interaction with an "intelligent" machine," *Int. J. Man-Mach. Stud.*, vol. 27, no. 5/6, pp. 479–525, Nov. 1987.

[17] R. Parasuraman and V. Riley, "Humans and automation: Use, misuse, disuse, abuse," *Human Factors*, vol. 39, no. 2, pp. 230–253, 1997.

[18] P. J. Smith, C. E. McCoy, and C. Layton, "Brittleness in the design of cooperative problem-solving systems: The effects on user performance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 27, no. 3, pp. 360–371, May 1997.

[19] S. A. Guerlain, P. J. Smith, J. H. Obradovich, S. Rudmann, P. Strohm, J. W. Smith, J. Svirbely, and L. Sachs, "Interactive critiquing as a form of decision support: An empirical evaluation," *Human Factors*, vol. 41, no. 1, pp. 72–89, Mar. 1999.

[20] D. A. Thurman and C. M. Mitchell, "An apprenticeship approach for the development of operations automation knowledge bases," in *Proc. 44th Annu. Meeting Human FactorsErgonom. Soc.*, 2000, pp. 231–234.

[21] D. A. Norman, "The problem with automation: Inappropriate feedback and interaction, not over-automation," *Phil. Trans. Roy. Soc. London. Ser. B, Biol. Sci.*, vol. 327, pp. 585–593, 1990.

[22] G. A. Jamieson and K. J. Vicente, "Designing effective human-automation plant interfaces: A control-theoretic perspective," *Human Factors*, vol. 47, no. 1, pp. 12–34, Spring 2005.

[23] N. B. Sarter and D. D. Woods, "How in the world did we ever get into that mode? Mode error and awareness in supervisory control," *Human Factors*, vol. 37, no. 1, pp. 5–19, Mar. 1995.

[24] E. Palmer, "'Oops, it didn't arm'—A case study of two automation surprises," in *Proc. 8th Int. Symp. Aviation Psychol.*, 1995, pp. 227–232.

[25] A. Degani, M. Shafto, and A. Kirlik, "Modes in human-machine systems: Review, classification, and application," *Int. J. Aviation Psychol.*, vol. 9, no. 2, pp. 125–138, 1999.

[26] K. Funk, C. Suroteguh, J. Wilson, and B. Lyall, "Flight deck automation and task management," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 1998, pp. 863–868.

[27] R. J. Mumaw, N. B. Sarter, and C. D. Wickens, "Analysis of pilots monitoring and performance on an automated flight deck," in *Proc. 11th Int. Symp. Aviation Psychol.*, 2001, [CD-ROM].

[28] N. B. Sarter, R. J. Mumaw, and C. D. Wickens, "Pilots' monitoring strategies and performance on automated flight decks: An empirical study combining behavioral and eye-tracking data," *Human Factors*, vol. 49, no. 3, pp. 347–357, Jun. 2007.

[29] T. B. Sheridan, "Toward a general model of supervisory control," in *Monitoring Behavior and Supervisory Control*, T. B. Sheridan and G. Johannsen, Eds. New York: Taylor & Fancis, 1976, pp. 271–282.

[30] D. D. Woods, "Cognitive demands and activities in dynamic fault management: Abductive reasoning and disturbance management," in *Human Factors in Alarm Design*, N. Stanton, Ed. Bristol, U.K.: Taylor & Francis, 1994, pp. 63–92.

[31] K. L. Mosier and L. J. Skitka, "Human decision makers and automated decision aids: Made for each other?" in *Automation and Human Performance: Theory and Applications*, R. Parasuraman and M. Mouloua, Eds. Philadelphia, PA: Lawrence Erlbaum Assoc., Inc., 1996, pp. 201–220.

[32] A. Blandford, R. Butterworth, and P. Curzon, "PUMA footprints: Linking theory and craft skill in usability evaluation," in *Proc. INTERACT*, 2001, pp. 577–584.

[33] M. D. Byrne and S. Bovair, "A working memory model of a common procedural error," *Cogn. Sci.*, vol. 21, no. 1, pp. 31–61, Jan. 1997.

[34] P. Curzon and A. Blandford, "From a formal user model to design rules," in *Proc. 9th Int. Workshop Interact. Syst. Design, Spec., Verification*, 2002, pp. 1–15.

[35] P. Curzon and A. Blandford, "Formally justifying user-centered design rules: A case study on post-completion errors," in *Proc. 4th Int. Conf. Integr. Formal Methods*, 2004, pp. 461–480.

[36] K. J. Vicente, *Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-based Work*. Philadelphia, PA: Lawrence Erlbaum Assoc., Inc., 1999.

[37] J. M. Schraagen, S. F. Chipman, and V. L. Shalin, *Cognitive Task Analysis*. Philadelphia, PA: Lawrence Erlbaum Assoc., Inc., 2000.

[38] B. Kirwan and L. K. Ainsworth, *A Guide to Task Analysis*. London, U.K.: Taylor & Francis, 1992.

[39] N. Mazaeva and A. M. Bisantz, "On the representation of automation using a work domain analysis," *Theor. Issues Ergonom. Sci.*, vol. 8, no. 6, pp. 509–530, Nov. 2007.

[40] D. B. Kaber, N. Segall, R. S. Green, K. Entzian, and S. Junginger, "Using multiple cognitive task analysis methods for supervisory control interface design in high-throughput biological screening processes," *Cogn., Technol. Work*, vol. 8, no. 4, pp. 237–252, Oct. 2006.

[41] J. A. Adams, C. M. Humphrey, M. A. Goodrich, J. L. Cooper, B. S. Morse, C. Engh, and N. Rasmussen, "Cognitive task analysis for developing unmanned aerial vehicle wilderness search support," *J. Cogn. Eng. Decision Making*, vol. 3, no. 1, pp. 1–26, Mar. 2009.

[42] L. Sherry, M. Medina, M. Feary, and J. Otiker, "Automated tool for task analysis of NextGen automation," in *Proc. Integr. Commun., Navig. Surveillance Conf.*, 2008, pp. 1–9.

[43] G. A. Jamieson, C. A. Miller, W. H. Ho, and K. J. Vicente, "Integrating task- and work domain-based work analyses in ecological interface design: A process control case study," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 6, pp. 887–905, Nov. 2007.

[44] G. Jamieson, "Ecological interface design for petrochemical process control: An empirical assessment," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 6, pp. 906–920, Nov. 2007.

[45] F. Paternò, "Model-based design of interactive applications," *Intelligence*, vol. 11, no. 4, pp. 26–38, Dec. 2000.

[46] M. L. Bolton, E. J. Bass, and J. Raymond Comstock, "Spatial awareness in synthetic vision systems: Using spatial and temporal judgments to evaluate texture and field of view," *Human Factors*, vol. 49, no. 6, pp. 961–974, Dec. 2007.

[47] M. L. Bolton and E. J. Bass, "Comparing perceptual judgment and subjective measures of spatial awareness," *Appl. Ergonom.*, vol. 40, no. 4, pp. 597–607, Jul. 2009.

[48] M. R. Endsley, "Measurement of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 65–84, 1995.

[49] D. B. Kaber and M. R. Endsley, "The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task," *Theor. Issues Ergonom. Sci.*, vol. 5, no. 2, pp. 113–153, Mar. 2004.

[50] E. J. Bass and A. R. Pritchett, "Human-automated judge learning: A methodology for examining human interaction with information analysis automation," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 759–776, Jul. 2008.

[51] M. D. Byrne and R. W. Pew, "A history and primer of human performance modeling," *Rev. Human Factors Ergonom.*, vol. 5, no. 1, pp. 225–263, Jun. 2009.

[52] B. E. John and D. E. Kieras, "Using GOMS for user interface design and evaluation: Which technique?" *ACM Trans. Comput. Human Interact.*, vol. 3, no. 4, pp. 287–319, Dec. 1996.

[53] J. M. Wing, "A specifier's introduction to formal methods," *Computer*, vol. 23, no. 9, pp. 8–23, Sep. 1990.

[54] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Commun. ACM*, vol. 5, no. 7, pp. 394–397, Jul. 1962.

[55] L. de Moura, B. Dutertre, and N. Shankar, "A tutorial on satisfiability modulo theories," in *Proc. 19th Int. Conf. Comput. Aided Verification*, 2007, pp. 20–36.

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

[57] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science*, J. Leeuwen, A. R. Meyer, M. Paterson, and D. Perrin, Eds. Cambridge, MA: MIT Press, 1990, ch. 16, pp. 995–1072.

[58] J. R. Burch, E. M. Clarke, D. L. Dill, J. Hwang, and K. L. McMillan, "Symbolic model checking: 1020 states and beyond," *Inf. Comput.*, vol. 98, no. 2, pp. 142–171, Jun. 1992.

[59] G. Holzmann and D. Peled, "An improvement in formal verification," in *Proc. 7th Int. Conf. Formal Descript. Tech.*, 1994, pp. 197–211.

[60] S. Graf and H. Saïdi, "Verifying invariants using theorem proving," in *Proc. 8th Int. Conf. Comput. Aided Verification*, 1996, pp. 196–207.

[61] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proc. 4th ACM SIGACT-SIGPLAN Symp. Principles Programm. Lang.*, 1977, pp. 238–252.

[62] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement for symbolic model checking," *J. ACM*, vol. 50, no. 5, pp. 752–794, Sep. 2003.

[63] T. A. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annu. IEEE Symp. Logic Comput. Sci.*, 1996, pp. 278–292.

[64] T. A. Henzinger, Z. Manna, and A. Pnueli, "Timed transition systems," in *Proc. REX Workshop*, 1991, pp. 226–251.

[65] D. L. Parnas, "On the use of transition diagrams in the design of a user interface for an interactive computer system," in *Proc. 24th Nat. ACM Conf.*, 1969, pp. 379–385.

[66] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Programm.*, vol. 8, no. 3, pp. 231–274, Jun. 1987.

[67] A. Degani and M. Heymann, "Formal verification of human-automation interaction," *Human Factors*, vol. 44, no. 1, pp. 28–43, Spring 2002.

[68] H. Thimbleby, *Press On: Principles of Interaction Programming*. Cambridge, MA: MIT Press, 2007.

[69] M. Harrison and D. Duke, "A review of formalisms for describing interactive behaviour," in *Proc. Workshop Softw. Eng. Human-Comput. Interact.*, 1995, pp. 49–75.

[70] A. Dix, M. Ghazali, S. Gill, J. Hare, and D. Ramduny-Ellis, "Physigrams: Modelling devices for natural interaction," *Formal Aspects Comput.*, vol. 21, no. 6, pp. 613–641, Nov. 2009.

[71] L. Sherry and J. Ward, "A formalism for the specification of operationally embedded reactive systems," in *Proc. 14th Digit. Avionics Syst. Conf.*, 1995, pp. 416–421.

[72] L. Sherry and M. Feary, "Improving the aircraft cockpit user-interface: Using rule-based expert system models," *PC AI*, vol. 15, no. 6, pp. 21–25, Nov. 2001.

[73] M. B. Dwyer, V. Carr, and L. Hines, "Model checking graphical user interfaces using abstractions," in *Proc. 6th Eur. Softw. Eng. Conf.*, 1997, pp. 244–261.

[74] M. B. Dwyer, O. T. Robby, and W. Visser, "Analyzing interaction orderings with model checking," in *Proc. 19th IEEE Int. Conf. Autom. Softw. Eng.*, 2004, pp. 154–163.

[75] N. Kamel and Y. Aït-Ameur, "A formal model for care usability properties verification in multimodal HCI," in *Proc. IEEE Int. Conf. Pervasive Serv.*, 2007, pp. 341–348.

[76] N. Kamel, Y. Aït-Ameur, S. A. Selouani, and H. Hamam, "A formal model to handle the adaptability of multimodal user interfaces," in *Proc. 1st Int. Conf. Ambient Media Syst.*, 2008, pp. 1–7.

[77] J. C. Campos and M. Harrison, "Formally verifying interactive systems: A review," in *Proc. 4th Int. Eur. Workshop Design, Spec., Verification Interact. Syst.*, 1997, pp. 109–124.

[78] J. C. Campos and M. D. Harrison, "Interaction engineering using the IVY tool," in *Proc. 1st ACM SIGCHI Symp. Eng. Interact. Comput. Syst.*, 2009, pp. 35–44.

[79] G. D. Abowd, H. Wang, and A. F. Monk, "A formal technique for automated dialogue development," in *Proc. 1st Conf. Des. Interact. Syst.*, 1995, pp. 219–226.

[80] F. Paternò, "Formal reasoning about dialogue properties with automatic support," *Interact. Comput.*, vol. 9, no. 2, pp. 173–196, Aug. 1997.

[81] J. C. Campos and M. D. Harrison, "Systematic analysis of control panel interfaces using formal tools," in *Proc. 15th Int. Workshop Design, Verification Spec. Interact. Syst.*, 2008, pp. 72–85.

[82] M. L. Bolton, "Validating human-device interfaces with model checking and temporal logic properties automatically generated from task analytic models," in *Proc. 20th Behav. Represent. Model. Simul. Conf.*, 2011, pp. 130–137.

[83] M. Feary, "Automatic detection of interaction vulnerabilities in an executable specification," in *Proc. 7th Int. Conf. Eng. Psychol. Cogn. Ergonom.*, 2007, pp. 487–496.

[84] A. Degani, A. Gellatly, and M. Heymann, "HMI aspects of automotive climate control systems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2011, pp. 1795–1800.

[85] K. Loer and M. D. Harrison, "An integrated framework for the analysis of dependable interactive systems (IFADIS): Its tool support and evaluation," *Autom. Softw. Eng.*, vol. 13, no. 4, pp. 469–496, 2006.

[86] N. G. Leveson, L. D. Pinnel, S. D. Sandys, and J. D. Reese, "Analyzing software specifications for mode confusion potential," in *Proc. Workshop Human Error Syst. Develop.*, 1997, [CD-ROM].

[87] R. W. Butler, S. P. Miller, J. N. Potts, and V. A. Carreño, "A formal methods approach to the analysis of mode confusion," in *Proc. 17th Digit. Avionics Syst. Conf.*, 1998, pp. C41/1–C41/8.

[88] J. C. Campos and M. Harrison, "Model checking interactor specifications," *Autom. Softw. Eng.*, vol. 8, no. 3, pp. 275–310, Aug. 2001.

[89] G. Lüttgen and V. Carreño, "Analyzing mode confusion via model checking," in *Proc. Theor. Pract. Aspects SPIN Model Check.*, 1999, pp. 120–135.

[90] A. Joshi, S. P. Miller, and M. P. E. Heimdahl, "Mode confusion analysis of a flight guidance system using formal methods," in *Proc. 22nd Digit. Avionics Syst. Conf.*, Oct. 2003, pp. 2.D.1-1–2.D.1–12.

[91] J. C. C. M. D. Harrison, "Modelling and analysing the interactive behaviour of an infusion pump," in *Proc. 4th Int. ECEASST*, Potsdam, Germany, 2011.

[92] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A new symbolic model verifier," in *Proc. 11th Int. Conf. Comput. Aided Verification*, 1999, pp. 682–688.

[93] S. Owre, J. Rushby, and N. Shankar, "PVS: A prototype verification system," in *Proc. 11th Int. Conf. Autom. Deduct.*, 1992, pp. 748–752.

[94] A. Degani, *Taming HAL: Designing Interfaces Beyond 2001*. New York: Macmillan, 2004.

[95] M. Oishi, I. Mitchell, A. Bayen, C. Tomlin, and A. Degani, "Hybrid verification of an interface for an automatic landing," in *Proc. 41st IEEE Conf. Decis. Control*, 2002, pp. 1607–1613.

[96] M. Oishi, I. Hwang, and C. Tomlin, "Immediate observability of discrete event systems with application to user-interface design," in *Proc. 42nd IEEE Conf. Decision Control*, 2003, pp. 2665–2672.

[97] L. Sherry, M. Feary, P. Polson, and E. Palmer, "Formal method for identifying two types of automation-surprises," Honeywell, Phoenix, AZ, Tech. Rep. C69-5370-016, 2000.

[98] J. Rushby, J. Crow, and E. Palmer, "An automated method to detect potential mode confusions," in *Proc. 18th Digit. Avionics Syst. Conf.*, 1999, pp. 4.B.2-1–4.B.2-6.

[99] J. Rushby, "Using model checking to help discover mode confusions and other automation surprises," *Rel. Eng. Syst. Safety*, vol. 75, no. 2, pp. 167–177, 2002.

[100] D. L. Dill, A. J. Drexler, A. J. Hu, and C. H. Yang, "Protocol verification as a hardware design aid," in *Proc. IEEE Int. Conf. Comput. Design—VLSI Comput. Process.*, 1992, pp. 522–525.

[101] B. Buth, "Analyzing mode confusion: An approach using FDR2," in *Proc. 23rd Int. Conf. Comput. Safety, Rel., Security*, 2004 pp. 101–114.

[102] *Formal Systems (Europe) Ltd, Failures-Divergence Refinement: FDR2 User Manual, Formal Syst. (Europe) Ltd., Oxford, U.K.*, , 2005, 6th ed. [Online]. Available: http://www.fsel.com/fdr2_manual.html

[103] E. J. Bass, K. M. Feigh, E. Gunter, and J. Rushby, "Formal modeling and analysis for interactive hybrid systems," in *Proc. 4th Int. ECEASST*, Potsdam, Germany, 2011.

[104] L. de Moura, H. Rueß, and M. Sorea, "Lazy theorem proving for bounded model checking over infinite domains," in *Proc. 18th Int. Conf. Autom. Deduct.*, 2002, pp. 438–455.

[105] J. Crow, D. Javaux, and J. Rushby, "Models and mechanized methods that integrate human factors into automation design," in *Proc. Int. Conf. Human-Comput. Interact. Aeronaut.*, 2000, pp. 163–168.

[106] M. Heymann and A. Degani, "Formal analysis and automatic generation of user interfaces: Approach, methodology, and an algorithm," *Human Factors*, vol. 49, no. 2, pp. 311–330, Apr. 2007.

[107] S. Combéfis and C. Pecheur, "A bisimulation-based approach to the analysis of human-computer interaction," in *Proc. ACM SIGCHI Symp. Eng. Interact. Comput. Syst.*, 2009, pp. 101–110.

[108] S. Combéfis, D. Giannakopoulou, C. Pecheur, and M. Feary, "A formal framework for design and analysis of human-machine interaction," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2011, pp. 1801–1808.

[109] J. Cañas, A. Antolí, and J. Quesada, "The role of working memory on measuring mental models of physical systems," *Psicológica*, vol. 22, no. 1, pp. 25–42, 2001.

[110] D. Norman, "Some observations on mental models," in *Mental Models*, D. Gentner and A. L. Stevens, Eds. Mahwah, NJ: Lawrence Erlbaum Assoc. Inc, 1983, pp. 7–14.

[111] J. Bredereke and A. Lankenau, "A rigorous view of mode confusion," in *Proc. 21st Int. Conf. Comput. Safety, Rel., Security*, 2002, pp. 19–31.

[112] J. Bredereke and A. Lankenau, "Safety-relevant mode confusions–modelling and reducing them," *Rel. Eng. Syst. Safety*, vol. 88, no. 3, pp. 229–245, 2005.

[113] P. H. Wheeler, "Aspects of automation mode confusion," M.S. thesis, MIT, Cambridge, MA, 2007.

[114] C. A. R. Hoare, "Communicating sequential processes," *Commun. ACM*, vol. 21, no. 8, pp. 666–677, Aug. 1978.

[115] J. Bredereke, "On preventing telephony feature interactions which are shared-control mode confusions," in *Feature Interactions in Telecommunications and Software Systems VII*. Lansdale, PA: IOS Press, 2003, pp. 159–176.

[116] D. Javaux, "A method for predicting errors when interacting with finite state systems. How implicit learning shapes the user's knowledge of a system," *Rel. Eng. Syst. Safety*, vol. 75, no. 2, pp. 147–165, Feb. 2002.

[117] F. Paternò, C. Mancini, and S. Meniconi, "ConcurTaskTrees: A diagrammatic notation for specifying task models," in *Proc. IFIP TC13 Int. Conf. Human-Comput. Interact.*, 1997, pp. 362–369.

[118] C. M. Mitchell and R. A. Miller, "A discrete control model of operator function: A methodology for information dislay design," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 16, no. 3, pp. 343–357, May 1986.

[119] M. L. Bolton, R. I. Siminiceanu, and E. J. Bass, "A systematic approach to model checking human-automation interaction using task-analytic models," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 5, pp. 961–976, Sep. 2011.

[120] H. R. Hartson, A. C. Siochi, and D. Hix, "The UAN: A useroriented representation for direct manipulation interface designs," *ACM Trans. Inf. Syst.*, vol. 8, no. 3, pp. 181–203, Aug. 1990.

[121] M. Giese, T. Mistrzyk, A. Pfau, G. Szwillus, and M. Detten, "AMBOSS: A task modeling approach for safety-critical systems," in *Proc. 2nd Conf. Human-Centered Softw. Eng. 7th Int. Workshop Task Models Diag.*, 2008, pp. 98–109.

[122] M. L. Bolton and E. J. Bass, "Formally verifying human-automation interaction as part of a system model: Limitations and tradeoffs," *Innov. Syst. Softw. Eng., A NASA J.*, vol. 6, no. 3, pp. 219–231, 2010.

[123] A. Degani, M. Heymann, and M. Shafto, "Formal aspects of procedures: The problem of sequential correctness," in *Proc. 43rd Annu. Meeting Human Factors Ergonom. Soc.*, 1999, pp. 1113–1117.

[124] S. Basnyat, P. Palanque, B. Schupp, and P. Wright, "Formal sociotechnical barrier modelling for safety-critical interactive systems design," *Safety Sci.*, vol. 45, no. 5, pp. 545–565, Jun. 2007.

[125] S. Basnyat, P. A. Palanque, R. Bernhaupt, and E. Poupart, "Formal modelling of incidents and accidents as a means for enriching training material for satellite control operations," in *Proc. Joint ESREL/17th SRA-Eur. Conf.*, 2008, [CD-ROM].

[126] E. L. Gunter, A. Yasmeen, C. A. Gunter, and A. Nguyen, "Specifying and analyzing workflows for automated identification and data capture," in *Proc. 42nd Hawaii Int. Conf. Syst. Sci.*, 2009, pp. 1–11.

[127] P. A. Palanque, R. Bastide, and V. Senges, "Validating interactive system design through the verification of formal task and system models," in *Proc. IFIP TC2/WG2.7 Work. Conf. Eng. Human-Comput. Interact.*, 1996, pp. 189–212.

[128] R. E. Fields, "Analysis of erroneous actions in the design of critical systems," Ph.D. dissertation, Univ. York, York, U.K., 2001.

[129] Y. Aït-Ameur, M. Baron, and P. Girard, "Formal validation of HCI user tasks," in *Proc. Int. Conf. Softw. Eng. Res. Pract.*, 2003, pp. 732–738.

[130] Y. Aït-Ameur and M. Baron, "Formal and experimental validation approaches in HCI systems design based on a shared event B model," *Int. J. Softw. Tools Technol. Transf.*, vol. 8, no. 6, pp. 547–563, 2006.

[131] F. Paternò and C. Santoro, "Integrating model checking and HCI tools to help designers verify user interface properties," in *Proc. 7th Int. Workshop Design, Spec., Verification Interact. Syst.*, 2001, pp. 135–150.

[132] P. V. Eijk and M. Diaz, Eds., *Formal Description Technique LOTOS: Results of the Esprit Sedos Project*. New York: Elsevier, 1989.

[133] L. De Moura, S. Owre, and N. Shankar, "The SAL language manual," Comput. Sci. Lab., SRI Int., Menlo Park, CA, Tech. Rep. CSL-01-01, 2003.

[134] M. L. Bolton and E. J. Bass, "A method for the formal verification of human interactive systems," in *Proc. 53rd Annu. Meeting Human Factors Ergonom. Soc.*, 2009, pp. 764–768.

[135] M. L. Bolton and E. J. Bass, "Using task analytic models to visualize model checker counterexamples," in *Proc. Int. Conf. Syst., Man, Cybern.*, 2010, pp. 2069–2074.

[136] F. Paternò, C. Santoro, and S. Tahmassebi, "Formal model for cooperative tasks: Concepts and an application for en-route air traffic control," in *Proc. 5th Int. Conf. Design, Spec., Verification Interact. Syst.*, 1998, pp. 71–86.

[137] E. J. Bass, M. L. Bolton, K. Feigh, D. Griffith, E. Gunter, W. Mansky, and J. Rushby, "Toward a multi-method approach to formalizing human-automation interaction and human-human communications," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2011, pp. 1817–1824.

[138] R. Bastide and S. Basnyat, "Error patterns: Systematic investigation of deviations in task models," in *Proc. 5th Int. Workshop Task Models Diagr. Users Interf. Design*, 2007, pp. 109–121.

[139] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Using formal methods to predict human error and system failures," in *Proc. 2nd Int. Conf. Appl. Human Factors Ergonom.*, 2008, [CD-ROM].

[140] M. L. Bolton and E. J. Bass, "Formal modeling of erroneous human behavior and its implications for model checking," in *Proc. 6th NASA Langley Formal Methods Workshop*, 2008, pp. 62–64.

[141] F. Paternò and C. Santoro, "Preventing user errors by systematic analysis of deviations from the system task model," *Int. J. Human-Comput. Stud.*, vol. 56, no. 2, pp. 225–245, Feb. 2002.

[142] E. Hollnagel, "The phenotype of erroneous actions," *Int. J. Man-Mach. Stud.*, vol. 39, no. 1, pp. 1–32, Jul. 1993.

[143] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Generating phenotypical erroneous human behavior to evaluate human-automation interaction using model checking," *Int. J. Human- Comput. Stud.*, 2012, to be published. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581912000997

[144] M. L. Bolton and E. J. Bass, "Evaluating human-automation interaction using task analytic behavior models, strategic knowledge based erroneous human behavior generation, and model checking," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2011, pp. 1788–1794.

[145] A. Cerone, P. A. Lindsay, and S. Connelly, "Formal analysis of human computer interaction using model-checking," in *Proc. 3rd IEEE Int. Conf. Softw. Eng. Formal Methods*, 2005, pp. 352–362.

[146] P. Lindsay and S. Connelly, "Modelling erroneous operator behaviours for an air-traffic control task," in *Proc. 3rd Australas. User Interf. Conf.*, 2002, vol. 7, pp. 43–54.

[147] A. Blandford, R. Butterworth, and J. Good, "Users as rational interacting agents: Formalising assumptions about cognition and interaction," in *Proc. 4th Int. Eurograph. Workshop, Design, Spec. Verification Interact. Syst.*, 1997, vol. 97, pp. 45–60.

[148] A. Blandford, R. Butterworth, and P. Curzon, "Models of interactive systems: A case study on programmable user modelling," *Int. J. Human-Comput. Stud.*, vol. 60, no. 2, pp. 149–200, Feb. 2004.

[149] R. Butterworth, A. Blandford, and D. Duke, "The role of formal proof in modelling interactive behaviour," in *Proc. 5th Int. Eurograph. Workshop Des., Spec. Verification Interact. Syst.*, 1998, pp. 87–101.

[150] R. Butterworth, A. Blandford, and D. Duke, "Demonstrating the cognitive plausibility of interactive system specifications," *Formal Aspects Comput.*, vol. 12, no. 4, pp. 237–259, 2000.

[151] R. M. Young, T. R. G. Green, and T. Simon, "Programmable user models for predictive evaluation of interface designs," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1989, pp. 15–19.

[152] R. Rukšėnas, P. Curzon, J. Back, and A. Blandford, "Formal modelling of cognitive interpretation," in *Proc. 13th Int. Workshop Des., Spec., Verification Interact. Syst.*, 2007, pp. 123–136.

[153] P. Curzon, R. Rukšėnas, and A. Blandford, "An approach to formal verification of human computer interaction," *Formal Aspects Comput.*, vol. 19, no. 4, pp. 513–550, Oct. 2007.

[154] R. Rukšėnas, J. Back, P. Curzon, and A. Blandford, "Formal modelling of salience and cognitive load," in *Proc. 2nd Int. Workshop Formal Methods Interact. Syst.*, 2008, pp. 57–75.

[155] R. Rukšėnas, J. Back, P. Curzon, and A. Blandford, "Verification-guided modelling of salience and cognitive load," *Formal Aspects Comput.*, vol. 21, no. 6, pp. 541–569, Nov. 2009.

[156] T. A. Basuki, A. Cerone, A. Griesmayer, and R. Schlatte, "Model checking user behaviour using interacting components," *Formal Aspects Comput.*, vol. 21, no. 6, pp. 571–588, 2009.

[157] P. Masci, P. Curzon, A. Blandford, and D. Furniss, "Modelling distributed cognition systems in PVS," in *Proc. 4th Int. Workshop EASST*, Potsdam, Germany, 2011.

[158] R. Rukšėnas and P. Curzon, "Abstract models and cognitive mismatch in formal verification," in *Proc. 4th Int. Workshop EASST*, Potsdam, Germany, 2011.

[159] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data flow programming language LUSTRE," *Proc. IEEE*, vol. 79, no. 9, pp. 1305–1320, Sep. 1991.

[160] P. A. Abdulla, J. Deneux, G. Stålmarck, H. Ågren, and O. Åkerlund, "Designing safe, reliable systems using scade," in *Proc. 1st Int. Symp. Leveraging Appl. Formal Methods*, 2006, pp. 115–129.

[161] R. Bastide, D. Navarre, and P. Palanque, "A tool-supported design framework for safety critical interactive systems," *Interact. Comput.*, vol. 15, no. 3, pp. 309–328, Jul. 2003.

[162] H. Huang, R. Rukšėnas, M. G. A. Ament, P. Curzon, A. L. Cox, A. Blandford, and D. Brumby, "Capturing the distinction between task and device errors in a formal model of user behaviour," in *Proc. 4th Int. Workshop EASST*, Potsdam, Germany, 2011.

[163] R. C. Boyatt and J. E. Sinclair, "A "lightweight formal methods" perspective on investigating aspects of interactive systems," in *Pre-Proc. 2nd Int. Workshop Formal Methods Interact. Syst.*, 2007, pp. 35–50.

[164] D. Jackson, "Lightweight formal methods," in *Proc. Int. Symp. Formal Methods Eur., Formal Methods Increasing Softw. Productiv.*, 2001, p. 1.

[165] S. Basnyat, N. Chozos, C. W. Johnson, and P. A. Palanque, "Incident and accident investigation techniques to inform model-based design of safety-critical interactive systems," in *Proc. Int. Workshop Des., Spec. Verification Interact. Syst.*, 2006, pp. 51–66.

[166] C. W. Johnson, "The formal analysis of human-computer interaction during accident investigations," in *Proc. Conf. People Comput. IX*, 1994, pp. 285–297.

[167] C. W. Johnson and A. J. Telford, "Extending the application of formal methods to analyse human error and system failure during accident investigations," *Softw. Eng. J.*, vol. 11, no. 6, pp. 355–365, Nov. 1996.

[168] C. W. Johnson, "The application of user modeling techniques to reason about the human contribution to major accidents," in *Proc. 7th Int. Conf. User Model.*, 1999, pp. 13–22.

[169] D. B. Kaber and J. M. Riley, "Adaptive automation of a dynamic control task based on secondary task workload measurement," *Int. J. Cogn. Ergonom.*, vol. 3, no. 3, pp. 169–187, 1999.

[170] B. Hilburn, P. G. Jorna, E. A. Byrne, and R. Parasuraman, "The effect of adaptive air traffic control (ATC) decision aiding on controller mental workload," in *Human-Automation Interaction: Research and Practice*, M. Mouloua and J. Koonce, Eds. Mahwah, NJ: Lawrence Erlbaum Assoc., Inc., 1997, pp. 84–91.

[171] M. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 32–64, 1995.

[172] J. Lee and N. Moray, "Trust, control strategies and allocation of function in human-machine systems," *Ergonomics*, vol. 35, no. 10, pp. 1243–1270, Oct. 1992.

[173] A. Hu, "Simulation vs. formal: Absorb what is useful; reject what is useless," in *Proc. 3rd Int. Haifa Verification Conf. Hardw. Softw.,Verification Testing*, 2008, pp. 1–7.

[174] J. Yuan, J. Shen, J. Abraham, and A. Aziz, "On combining formal and informal verification," in *Proc. 9th Int. Conf. Comput. Aided Verification*, 1997, pp. 376–387.
[175] B. E. John, *CogTool User Guide*. Pittsburgh, PA: Carnegie Mellon Univ.
[176] J. R. Anderson, M. Matessa, and C. Lebiere, "ACT-R: A theory of higher level cognition and its relation to visual attention," *Human-Comput. Interact.*, vol. 12, no. 4, pp. 439–462, 1997.
[177] R. Rukšėnas, P. Curzon, A. Blandford, and J. Back, "Combining human error verification and timing analysis," in *Proc. Conf. Eng. Interact. Syst.*, 2009, pp. 18–35.

Matthew L. Bolton (S'05–M'10) received the B.S. degree in computer science, the M.S. and Ph.D. degrees in systems engineering from the University of Virginia, Charlottesville, VA.

Currently, he is an Assistant Professor with the Department of Mechanical and Industrial Engineering at the University of Illinois at Chicago. His research interests include human-automation interaction, human-performance modeling, systems engineering, task analytic human behavior models, and the applications of formal methods within these domains.

Ellen J. Bass (M'98–SM'03) received the B.S.Eng. and B.S.Econ. degrees from the University of Pennsylvania, Philadelphia, PA, the M.S. degree from the State University of New York at Binghamton, and the Ph.D. degree from the Georgia Institute of Technology, Atlanta, GA.

Currently, she is a Professor with the College of Information Science and Technology and College of Nursing and Health Professions, Drexel University, Philadelphia, PA. Her research focuses on characterizing human judgment and decision making, modeling human judgment when supported by information automation, and computational models of human-human and human-automation interaction.

Radu I. Siminiceanu received the B.S. and M.S. degrees in computer science from the University of Iaßi, Iaßi, Romania, and the Ph.D. degree in computer science from the College of William and Mary, Williamsburg, VA, in 2003.

Currently, he is a Senior Research Scientist at the National Institute of Aerospace, Hampton, VA, USA. His research interests include formal methods, in particular model checking, applied to aerospace, air traffic management, and avionics systems.