Using Task Analytic Models and Phenotypes of Erroneous Human Behavior to Discover System Failures Using Model Checking

Matthew L. Bolton and Ellen J. Bass
Department of Systems and Information Engineering
University of Virginia
Charlottesville, VA

Breakdowns in complex systems often occur as a result of system elements interacting in ways unanticipated by analysts or designers. In systems with human operators, human-automation interaction associated with both normative and erroneous human behavior can contribute to such failures. This paper presents a method for automatically generating task analytic models encompassing both erroneous and normative human behavior from normative task models. The resulting model can be integrated into a formal system model so that system safety properties can be formally verified with a model checker. This allows analysts to prove that a human automation-interactive system (as represented by the model) will or will not satisfy safety properties with both normative and generated erroneous human behavior. This method is illustrated with a case study: the operation of a radiation therapy machine. In this example, a problem resulting from a generated erroneous human action is discovered. Future extensions of our method are discussed.

INTRODUCTION

Complex, safety-critical systems involve the interaction of automated devices and goal-oriented human operators in a dynamic environment. Human-automation interaction (HAI), and particularly erroneous human behavior, can contribute to their failure (Reason, 1990; Hollnagel, 1993b). Two research areas, HAI and formal methods, have attempted to address these problems from different directions. HAI researchers investigate the way human operators interact with automation for the purpose of designing systems that facilitate safe, human work through the use of field studies, design principles, controlled experiments, and modeling and simulation analyses (Sheridan & Parasuraman, 2005). Formal methods researchers use well defined mathematical modeling and proof techniques to verify that system models do or do not exhibit desired properties (Wing, 1990). Model checking (Clarke, Grumberg, & Peled, 1999) is a type of formal verification that searches a model's entire statespace in order to find violations of properties written in temporal logic. The model checker will indicate if the specification is valid. Otherwise, it returns a counterexample: an execution trace illustrating how the specification was violated.

HAI analyses account for human behavior but risk missing potentially dangerous system states or interactions. Formal verification can guarantee that all modeled conditions will be analyzed during formal verification of safety properties and other invariants, but only for models that can fit in the memory of the computer being used for the analysis (Clarke et al., 1999).

In order to exploit the advantages offered by both domains, researchers have used formal verification to:

1. Evaluate human-device interfaces (HDIs) (Abowd, Wang, & Monk, 1995; Campos & Harrison, 2008);
2. Discover and eliminate potential mode confusion (Degani & Heymann, 2002; Bredereke & Lankenau, 2005);
3. Verify the safe operation of systems with cognitively plausible behavior using cognitive modeling (Lindsay & Connelly, 2002; Curzon, Rukšėnas, & Blandford, 2007); and
4. Verify the safety of task analytic models (Aït-Ameur & Baron, 2006; Fields, 2001).

See Bolton (2010) for an in-depth survey of these techniques.

Prior work manually incorporated erroneous human behaviors into task analytic models (Fields, 2001; Bastide & Basnyat, 2007). We show that erroneous human behavior can be automatically incorporated into task analytic models and that the resulting models can be used to formally verify system safety properties. We discuss the relevant literature on erroneous human behavior and show how erroneous human behavior generation can be integrated into our existing method (Bolton & Bass, 2009b) which has previously been used to verify systems with normative task analytic models (discussed below). We illustrate this process with a radiation therapy machine example.

Formal Verification of HAI

Our method (Bolton & Bass, 2009b) utilizes a formal modeling architectural framework which encompasses models of human missions (i.e. goals), human task behavior, HDIs, device automation, and the operational environment (Bolton & Bass, 2010). Human task behavior models are created using a task modeling language called Enhanced Operator Function Model (EOFM) (Bolton & Bass, 2009a; Bolton, Siminiceanu, & Bass, n.d.) which extends the Operator Function Model (OFM) (Mitchell & Miller, 1986). EOFMs are hierarchical and heterarchical representations of goal-driven activities that decompose into lower level activities, and, finally, atomic actions. EOFMs express task knowledge by explicitly specifying the conditions under which human operator activities can execute (preconditions), repeat (repeat conditions), and complete (completion conditions). Any activity can decompose into one or more other activities or one or more actions. A decomposition operator specifies the temporal relationships between and the cardinality of the decomposed activities or actions (when they can execute relative to each other and how many can execute).

EOFMs can be represented visually as a tree-like graph (see Figure 4). Actions are rectangles and activities are rounded rectangles. An activity's decomposition is presented as an arrow, labeled with the decomposition operator, that points to a large rounded rectangle containing the decomposed activities or actions. In the work presented here, three decomposition operators are used: (a) *ord* (all activities or actions in the decomposition must execute in the order they appear); (b) *or_seq* (one or more of the activities or actions in the decomposition must

execute); and (c) *xor* (exactly one activity or action in the decomposition must execute). Conditions on activities are represented as shapes or arrows (annotated with the condition logic) connected to the activity that they constrain. A precondition is a yellow, downward-pointing triangle; a completion condition is a magenta, upward-pointing triangle; and a repeat condition is an arrow recursively pointing to the top of the activity.

When used as part of a formal system model, instantiated EOFM task models are translated into the language of the Symbolic Analysis Laboratory (SAL) (De Moura, Owre, & Shankar, 2003) using the language's formal semantics (Bolton et al., n.d.). Formal verifications are performed on the complete system model using SAL's symbolic model checker (SAL-SMC). Any produced counterexamples can be visualized and evaluated using EOFM's visual notation (Bolton & Bass, in press).

Erroneous Human Behavior

Erroneous human behaviors have been classified based on how they arise from human decision making, cognition, and task execution (Jones, 1997). Instantiated EOFMs do not model low level perceptual, motor, and cognitive processes. However, they do model human task behavior hierarchically down to the atomic, observable action level. Thus they are compatible with Hollnagel's (1993b) phenotypes of erroneous action.

Hollnagel (1993b) classified erroneous human behavior based on its observable manifestation as divergences from planned or normative sequences of actions. All erroneous behaviors are composed of one or more erroneous actions, all capable of being detected by observing the performance of a single act in a plan. These "zero-order" phenotypes include: prematurely starting an action, delaying the start of an action, prematurely finishing an action, delaying the completion of an action, omitting an action, jumping forward (performing an action that should be performed later), jumping backward (performing a previously performed action), repeating an action, and performing an unplanned action (an intrusion). Higher order phenotypes are composed of at least two zero-order phenotypes.

The phenotypes for delays and premature starts and finishes specifically refer to time, not currently supported by our method (Bolton & Bass, 2009b). However, all of the other zero-order phenotypes relate to the performance or non-performance of actions; all of which are compatible with the formal semantics and structure of the EOFM. In this work, we introduce a method that automatically generates these phenotypes as part of an instantiated EOFM that captures normative behavior.

AUTOMATIC ERRONEOUS HUMAN BEHAVIOR GENERATION

The erroneous behavior generation process must be capable of replicating Hollnagel's (1993b) zero-order phenotypes for omitting, jumping to, repeating, or intruding an action for each original action in an instantiated EOFM. To allow for more complex erroneous human behaviors, the generation process must allow zero-order erroneous acts to be chained together. An unconstrained number of erroneous acts could result in an unbounded human task behavior model which would negate the explanatory power afforded by using task models. Thus, the erroneous behavior generation process must be able to constrain the number of erroneous acts that can be performed in the formally translated, erroneous human behavior model. In order to facilitate analysis, the EOFM to SAL translation process, and counterexample interpretation; the erroneous behavior generation structure should be represented in the EOFM language.

To generate erroneous actions, zero-order phenotypes for omissions, skips, jumps, repetitions, and intrusions are incorporated into an instantiated EOFM by replacing each atomic action (*Action*^{x}) with a customized structure (*Action*^{x}') (Figure 1). This design includes an upper bound on the number of erroneous acts (*EMax*) and a variable (*ECount*) that keeps track of the number of erroneous acts that the task model has performed. Any activity that represents an erroneous act has a precondition asserting that it can only be executed if the current number of performed erroneous actions is less than the maximum (*ECount* < *EMax*). Every time an activity representing an erroneous act executes, *ECount* is incremented by one (*ECount*++).

Action^x' decomposes into several additional activities, allowing Action^x' to complete execution if one or more of these activities executes (the *or_seq* decomposition operator). *CorrectAction* allows the original correct action (*Action*^{x}) to be performed. The *Omission* activity allows an operator to perform the erroneous act of omitting the original action, represented as the *DoNothing* action. The *Commission* activities each allow a single erroneous action to be performed (the *xor* decomposition operator), where the set of erroneous actions corresponds to the *n* human actions available to the human operator. There are *ECount Commission* activities, thus allowing up to *ECount* erroneous actions to occur in place of the original action.

This design allows the specified zero-order erroneous behavior phenotypes to be generated when the EOFM executes. Omissions occur through the *Omission* activity. Repetitions, forward or backward jumps, or intrusions can occur by executing either the current action, another planned action, or some other action respectively through one of the *Commission* activities. Multiple erroneous actions can occur instead of, before, or after the correct action due to the *or_seq* decomposition. Multiple erroneous acts can also occur between erroneous behavior generating structures for different actions. Thus, the use of this structure allows for single erroneous actions as well as more complicated erroneous behaviors to be generated.

Our Java-based EOFM to SAL translator (Bolton et al., n.d.) was modified to optionally incorporate erroneous behavior into any instantiated EOFM. The translator takes the maximum number of erroneous acts (*EMax*) as input and traverses the EOFM structure, replacing each action with its corresponding erroneous behavior generative structure (Figure 1). The translator represents *EMax* as a constant. It modifies each human operator by adding a local variable representing the current number of performed erroneous acts (*ECount*, which can assume a value from 0 to *EMax*) and a *DoNothing* human action.

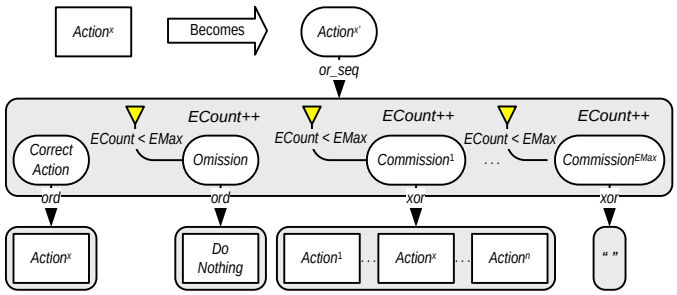

Figure 1. Visualization of the EOFM structure used to generate a zero-order phenotypical erroneous human behavior.

The translator produces two files as output. The first is an EOFM XML file representing the created erroneous human behavior model (separate from the model of normative behavior). The second is the translated SAL representation of this model.

APPLICATION

To demonstrate the erroneous behavior generation method, we evaluate a model of a human operated radiation therapy machine. This device is a room-sized, computer-controlled, medical linear accelerator. It has two treatment modes: the electron beam mode is used for shallow tissue treatment, and the xray mode is used for deeper treatments - requiring electron beam current approximately 100 times greater than that used for the electron beam mode. The xray mode uses a beam spreader (not used in electron beam mode) to produce a uniform treatment area and attenuate the radiation of the beam. The machine is capable of delivering a lethal dose of radiation to a patient if an xray treatment is administered without the spreader in place.

Human-device Interface

The HDI model (Figure 2) encompasses the behavior of the interface for selecting and administering treatments. It represents the information on a computer monitor in response to five relevant keyboard keys ('X', 'E', enter, up and 'B'). The interface states (*InterfaceState*) starts in *Edit* where the human operator can press 'X' or 'E' (*PressX* or *PressE*) to select the xray or electron beam mode and transition to the *ConfirmXrayData* or *ConfirmEBeamData* states respectively. When in the *ConfirmXrayData* or *ConfirmEBeamData* states, the appropriate treatment data is displayed (*DisplayedData*), and the human operator can confirm the displayed data by pressing enter (advancing to the *PrepareToFireXray* or *PrepareToFireEBeam* states) or return to the *Edit* state by pressing up (*PressUp*) on the keyboard. In the *PrepareToFireXray* or *PrepareToFireEBeam* states, the human operator must wait for the beam to become ready (*BeamState*), at which point he can press 'B' (*PressB*) to administer treatment by firing the beam. This transitions the interface state to *TreatmentAdministered*. The operator can also return to the previous data confirmation state by pressing up.

Device Automation

The device automation model (Figure 3) encompasses system behavior related to the power level of the beam (*BeamLevel*), the position of the spreader (*Spreader*), and the firing of the beam (*BeamFireState*). The power level of the beam (*BeamLevel*) is initially not set (*NotSet*). When the human operator selects the xray or electron beam treatment mode, the power level transitions to the appropriate setting (*XrayLevel* or *EBeamLevel* respectively). However, if the human operator selects a new power level, there is a delay in the transition to the correct power level, where it remains in an intermediary state (*XtoE* or *EtoX*) at the old power level before automatically transitioning to the new one. The position of the spreader (*Spreader*) starts either in or out of place (*InPlace* or *OutOfPlace*). When the human operator selects the xray or electron beam treatment mode, the spreader transitions to the appropriate setting (*InPlace* or *OutOfPlace* respectively). The firing state of the beam (*BeamFireState*) is initially waiting to be fired (*Waiting*). When the human operator fires the beam (pressing 'B' when the beam is ready), the beam fires (*Fired*) and returns to its waiting state.

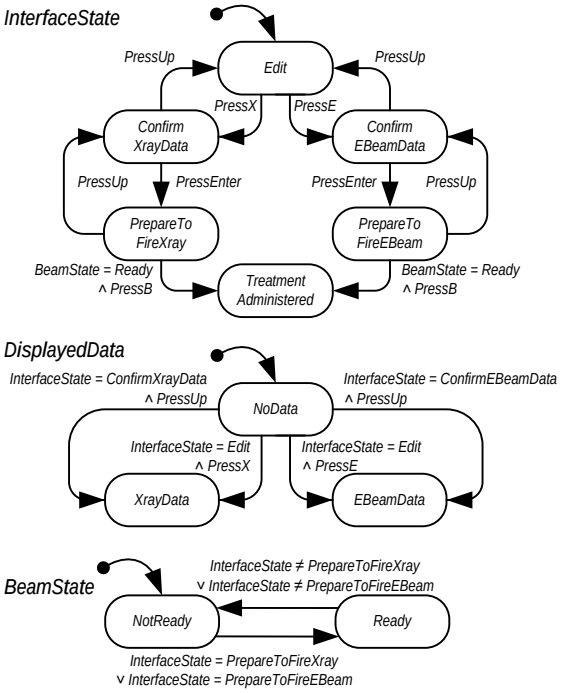

Figure 2. State transition representation of the formal human-device interface model. Rounded rectangles represent states. Arrows between states represent transitions. Initial states are pointed to by arrows starting with a dot.

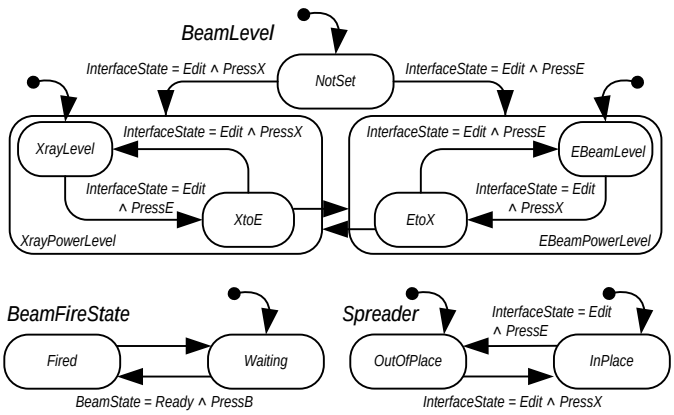

Figure 3. State transition representation of the formal device automation model.

Human Mission

The human mission identifies the desired treatment (*TreatmentType* equaling *Xray* or *EBeam*).

Human Task Behavior

Three goal directed task models describe the administration of treatment with the radiation therapy machine (Figure 4): selecting the treatment mode (*aSelectXorE*), confirming treatment data (*aConfirm*), and firing the beam (*aFire*).

These models access input variables from the HDI (the interface state (*InterfaceState*), the displayed treatment data (*DisplayedData*), and the ready status of the beam (*BeamState*)) and the mission (treatment type (*TreatmentType*)) to generate the human actions for pressing the appropriate keyboard keys.

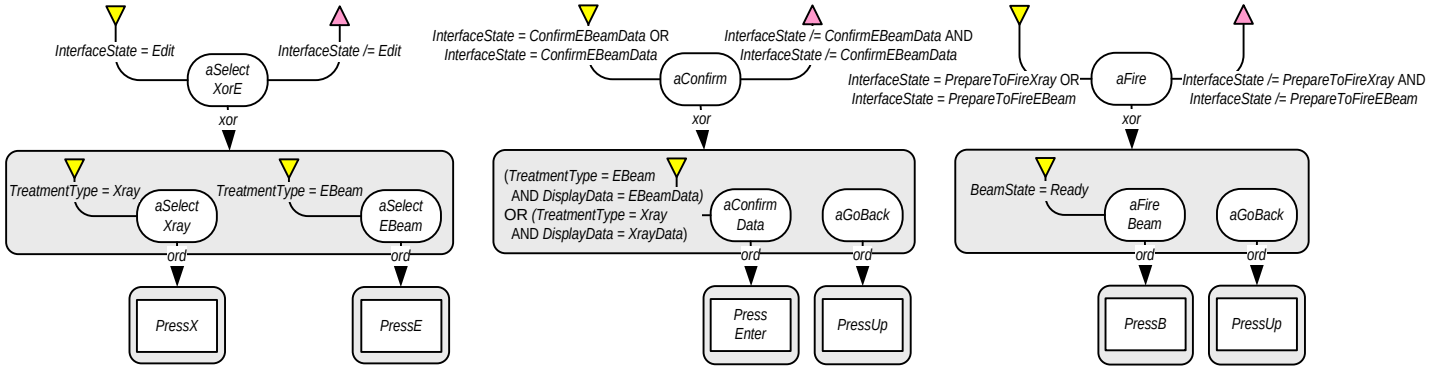

Figure 4. Visualization of the EOFMs for interacting with the radiation therapy machine: selecting the treatment mode (*aXorE*), confirming treatment data (*aConfirm*), and firing the beam (*aFire*).

The practitioner can select the treatment mode (*aSelectXorE*) when the interface is in the *Edit* state by performing either the *PressX* or *PressE* actions based on the mission *TreatmentType*. The practitioner can choose whether or not to confirm displayed treatment data (*aConfirm*) when the interface is in either of the two data confirmation states. If the displayed data corresponds to the desired treatment mode, the practitioner confirms it with a *PressEnter* action. He or she can return to the *Edit* state via a *PressUp* action. The practitioner decides whether or not to administer treatment (*aFire*) when the interface is in either of the states for preparing to fire the beam. He or she can fire the beam (if the beam is ready) by pressing 'B' (*PressB*) or return to the previous interface state by pressing up (*PressUp*).

EOFM to SAL Translation

The EOFM instance was translated twice into SAL code and incorporated into the larger formal system model: once with normative behavior and once with erroneous human behavior with a maximum of one erroneous act. The normative behavior model's EOFM representation was 74 lines of code. Its corresponding formal representation was 166 lines of SAL code. The produced erroneous EOFM model was 240 lines of code. Its formal representation was 641 lines of SAL code.

Specification and Verification

We specify that we never want the radiation therapy machine to irradiate a patient by administering an unshielded xray treatment using Linear Temporal Logic as follows:

$$\mathbf{G}\neg\left(\begin{array}{l} BeamFireState = Fired \\ \wedge\ BeamLevel = XrayPowerLevel \\ \wedge\ Spreader = OutOfPlace \end{array}\right) \qquad (1)$$

When checked against the formal system model with the translated normative task behavior, this verified to true in less than one second having visited 1648 states.

The formal system model containing the erroneous human behavior (which had 45290 states) produced a counterexample after 38 seconds illustrating the following failure sequence:

1. The model initialized with the interface in the edit state with no displayed data and the beam not ready; the beam power level not set; the spreader out of place; the beam fire state waiting; and the human mission indicating that the human operator should administer electron beam treatment.
2. When attempting to select the electron beam mode, the practitioner erroneously pressed 'X' instead of 'E' (via a generated *Commission* activity in *aSelectXorE* from Figure 4). This caused the interface to transition to the xray data confirmation state and display the xray treatment data. The spreader was also moved in place and the beam was set to the xray power level.
3. Because the incorrect data was displayed, the practitioner pressed up to return the interface to the edit mode.
4. The practitioner selected electron beam treatment mode by pressing the 'E' key. The interface transitioned to the electron beam data confirmation state and displayed the electron beam treatment data. The spreader was moved out of place and the beam prepared to transition to the electron beam power level (XtoE in Figure 4).
5. The practitioner confirmed the treatment data by pressing enter and the interface transitioned to the electron beam's waiting to fire state.
6. The beam became ready.
7. The practitioner fired the beam by pressing 'B'. Because the beam power level had not yet transitioned to the electron beam power level, the beam fired at the xray power level with the spreader out of place.

DISCUSSION

The generation process satisfies all of our goals for including erroneous human behavior in the formal verification of HAI:

1. Using an EOFM task structure to replace every action in an instantiated EOFM, we are capable of generating zero-order phenotypes of erroneous human action: omitting an action, jumping to another planned action, repeating the last performed action, or intruding an action.
2. The use of the structure in each action, and the use of the *or_seq* decomposition operator in the erroneous behavior generation structure allow multiple zero-order phenotypical erroneous acts to generate all of Hollnagel's first-order phenotypes except for time compression.
3. The number of possible erroneous acts is constrained by a maximum and a counter preventing the task behavior model from becoming unbounded.
4. The erroneous behavior generation structure is represented in EOFM constructs and is thus compatible with the EOFM to SAL translator and counterexample visualizer.

The radiation therapy machine example illustrates how this process can be used to find potential system failures associated

with erroneous human behavior. The discovered problem is very similar to real problems found in radiation therapy machines which have resulted in patient injury and death (Leveson & Turner, 1993; Bogdanich, 2010).

Benchmarking

The erroneous behavior generation process increases the amount of structure necessary to formally represent the task behavior and the amount of structure increases with the number of allowable erroneous behaviors. Given that formal verification is limited by the size of the system model, the use of erroneous human behavior generation may be limited by the size of the target model. Future work should characterize how erroneous behavior generation impacts model statespace complexity.

Optimization

In the current implementation there are situations where non-erroneous acts are counted as erroneous. For example, the correct action could be executed through a *Commission* activity without the *CorrectAction* activity executing. Similarly, the current structure allows for the same erroneous acts to be performed in multiple ways. For example, a given erroneous act could be performed through different *Commission* activities. The structure used to generate erroneous human behavior should be optimized to eliminate such redundancies.

Erroneous Human Behavior Generation

Our method currently does not support Hollnagel's (1993b) phenotypes for prematurely starting or finishing an action, or delaying the start or completion of an action. Future work should investigate if these types of erroneous behaviors can be automatically generated as an extension of our method. In addition, there are cognitive reasons why specific sequences of erroneous behavior phenotypes may be more likely than others (Reason, 1990). Future work should determine if it is possible to use cognitive principals to automatically generate erroneous behaviors from normative task analytic behavior models.

Comparison with Other Methods

Our method supports the generation of erroneous human behavior comparable to that generated using formal models of cognition (Curzon et al., 2007). Future work should investigate if there are any discrepancies between the erroneous behaviors generated by this technique and ours, and determine if there are applications better suited to analyses with either approach.

Human Reliability Analysis (HRA) (Hollnagel, 1993a) is a more established system evaluation technique which allows analysts to assess the probability of failures in light of erroneous human behavior. Future work should determine what the advantages and disadvantage of HRA are compared to our technique.

ACKNOWLEDGEMENT

This work was supported in part by National Library of Medicine (NLM) grant T15LM009462 and Research Grant Agreement UVA-03-01, sub-award 6073-VA and 2723-VA from the National Institute of Aerospace (NIA). The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIA, NASA, NLM, or NIH. The authors thank Radu Siminiceanu of the NIA and Ben Di Vito of the NASA Langley Research Center for technical help.

REFERENCES

Abowd, G. D., Wang, H., & Monk, A. F. (1995). A formal technique for automated dialogue development. In *Proceedings of the 1st Conference on Designing Interactive Systems* (pp. 219–226). New York: ACM.

Aït-Ameur, Y., & Baron, M. (2006). Formal and experimental validation approaches in HCI systems design based on a shared event B model. *International Journal on Software Tools for Technology Transfer*, *8*(6), 547–563.

Bastide, R., & Basnyat, S. (2007). Error patterns: Systematic investigation of deviations in task models. In *Task models and diagrams for users interface design* (pp. 109–121). Berlin: Springer.

Bogdanich, W. (2010, January 23). The radiation boom: Radiation offers new cures, and ways to do harm. *The New York Times*.

Bolton, M. L. (2010). *Using task analytic behavior modeling, erroneous human behavior generation, and formal methods to evaluate the role of human-automation interaction in system failure*. Unpublished doctoral dissertation, University of Virginia, Charlottesville.

Bolton, M. L., & Bass, E. J. (2009a). Enhanced operator function model: A generic human task behavior modeling language. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics* (pp. 2983–2990). Piscataway: IEEE.

Bolton, M. L., & Bass, E. J. (2009b). A method for the formal verification of human interactive systems. In *Proceedings of the 53rd Annual Meeting of the Human Factors and Ergonomics Society* (pp. 764–768). Santa Monica: Human Factors and Ergonomics Society.

Bolton, M. L., & Bass, E. J. (2010). Formally verifying human-automation interaction as part of a system model: Limitations and tradeoffs. *Innovations in Systems and Software Engineering: A NASA Journal.* doi : 10.1007/s11334 – 010 – 0129 – 9.

Bolton, M. L., & Bass, E. J. (in press). Using task analytic models to visualize model checker counterexamples. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics.* Piscataway: IEEE.

Bolton, M. L., Siminiceanu, R. I., & Bass, E. J. (n.d.). A systematic approach to model checking human-automation interaction using task-analytic models. Under review.

Bredereke, J., & Lankenau, A. (2005). Safety-relevant mode confusions–modelling and reducing them. *Reliability Engineering and System Safety*, *88*(3), 229–245.

Campos, J. C., & Harrison, M. D. (2008). Systematic analysis of control panel interfaces using formal tools. In *Proceedings of the 15th International Workshop on the Design, Verification and Specification of Interactive Systems* (pp. 72–85). Berlin: Springer.

Clarke, E. M., Grumberg, O., & Peled, D. A. (1999). *Model checking*. Cambridge: MIT Press.

Curzon, P., Rukšėnas, R., & Blandford, A. (2007). An approach to formal verification of humanŰcomputer interaction. *Formal Aspects of Computing*, *19*(4), 513–550.

Degani, A., & Heymann, M. (2002). Formal verification of human-automation interaction. *Human Factors*, *44*(1), 28–43.

De Moura, L., Owre, S., & Shankar, N. (2003). *The SAL language manual* (Tech. Rep. No. CSL-01-01). Menlo Park: Computer Science Laboratory, SRI International.

Fields, R. E. (2001). *Analysis of erroneous actions in the design of critical systems*. Unpublished doctoral dissertation, University of York, York.

Hollnagel, E. (1993a). *Human reliability analysis: Context and control*. Academic Press.

Hollnagel, E. (1993b). The phenotype of erroneous actions. *International Journal of Man-Machine Studies*, *39*(1), 1–32.

Jones, P. M. (1997). Human error and its amelioration. In *Handbook of systems engineering and management* (pp. 687–702). Malden: Wiley.

Leveson, N. G., & Turner, C. S. (1993). An investigation of the therac-25 accidents. *Computer*, *26*(7), 18–41.

Lindsay, P., & Connelly, S. (2002). Modelling erroneous operator behaviours for an air-traffic control task. In *Proceedings of the 3rd Australasian User Interface Conference* (Vol. 7, p. 43-54). Melbourne: ACS.

Mitchell, C. M., & Miller, R. A. (1986). A discrete control model of operator function: A methodology for information dislay design. *IEEE Transactions on Systems Man Cybernetics Part A: Systems and Humans*, *16*(3), 343–357.

Reason, J. (1990). *Human error*. New York: Cambridge University Press.

Sheridan, T. B., & Parasuraman, R. (2005). Human-automation interaction. *Reviews of human factors and ergonomics*, *1*(1), 89–129.

Wing, J. M. (1990). A specifier's introduction to formal methods. *Computer*, *23*(9), 8, 10–22, 24.