REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
13-11-2017	Final Report	15-Aug-2015 - 14-Aug-2017

4. TITLE AND SUBTITLE

Final Report: Young Investigator Program (8.5): Preventing Complex Failures of Human Interactive Systems with Erroneous Behavior Generation and Robust Human Task Behavior Patterns

5a. CONTRACT NUMBER
W911NF-15-1-0474

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES

State University of New York (SUNY) at Bı
The UB Commons
520 Lee Entrance, Suite 211
Amherst, NY 14228 -2567

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
67017-LS-YIP.2

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Matthew Bolton
UU	UU	UU	UU		19b. TELEPHONE NUMBER: 716-645-2359

RPPR Final Report
as of 17-Nov-2017

Agency Code:

Proposal Number: 67017LSYIP **Agreement Number: W911NF-15-1-0474**
INVESTIGATOR(S):

Name: Ph.D. Matthew Lee Bolton
Email: mbolton@buffalo.edu
Phone Number: 7166452359
Principal: Y

Organization: **State University of New York (SUNY) at Buffalo**
Address: The UB Commons, Amherst, NY 142282567
Country: USA
DUNS Number: 038633251 EIN: 141368361
Report Date: 14-Nov-2017 Date Received: 13-Nov-2017
Final Report for Period Beginning 15-Aug-2015 and Ending 14-Aug-2017
Title: Young Investigator Program (8.5): Preventing Complex Failures of Human Interactive Systems with Erroneous Behavior Generation and Robust Human Task Behavior Patterns
Begin Performance Period: 15-Aug-2015 **End Performance Period:** 14-Aug-2017
Report Term: 0-Other
Submitted By: Ph.D. Matthew Bolton Email: mbolton@buffalo.edu
Phone: (716) 645-2359
Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: 1 **STEM Participants:** 2

Major Goals: The research objectives of this project were to be accomplished through the completion of two goals. Goal 1 was to develop and evaluate a novel taxonomy of erroneous human behavior. Goal 2 was to adapt this taxonomy to a formal erroneous behavior generation method and model checking process and test them by applying them to case studies. Each goal is discussed in more detail below.

Goal 1 was performed over the first year of the project (August 2015 - August 2016): The purpose of Goal 1 was to define a new erroneous behavior taxonomy based on where and how erroneous behaviors can occur as deviations from human task models due to failures of attention. These can occur due to either environmental criteria or the task itself. This taxonomy accounted for a human's failure to perform activities or actions, the repetition of activities or actions, spurious performance of actions or activities from other tasks, and spurious performance of actions or activities in the current task. Because the recovery behavior humans may use to account for erroneous acts can also be considered erroneous (it deviates from the normative task) and can be as important to system safety as the initiating erroneous act, the taxonomy also incorporates the different ways that it can manifest. This taxonomy is compatible with Reason's genotypical slips because a behavior's point of deviation from a task indicate what element of the task or environment the human failed to attend to. It is also compatible with Hollnagel's phenotypes of erroneous action in that it allows for all phenotypes to be replicable as deviations from task. Once the taxonomy was formally defined, we showed that it encompassed the desired behaviors and that it was capable of subsuming both Hollnagel's and Reason's classifications without any loss of information.

Goal 2 was performed over the second year of the project (August 2016 - August 2017): Under goal 2, we adapted the new taxonomy into a formal erroneous behavior generation approach. This approach extended existing EOFM-based erroneous behavior generation to allow deviations from task models in all of the ways dictated by the taxonomy. This generation method was incorporated into EOFM-supported model checking analyses. To demonstrate the ability of this new method to discover situations where erroneous behavior can contribute to failures, we have showed that it can find known failures discovered in older analyses while offering additional analysis capabilities. We have also applied the method to new, Army-relevant case studies. First, we applied the method to the operation of an Army unmanned aerial system when remote pilots are attempting to deal with fuel shortages across fuel tanks (a situation that can lead to a crash if not properly addressed). Finally, we applied the method to the operation of an Army Apache helicopter when identifying and engaging with potentially hostile targets (a situation where mistakes could lead to fratricide).

Accomplishments: Goal 1

RPPR Final Report
as of 17-Nov-2017

Task 1.1: Taxonomy development
Task 1.1 defined our erroneous behavior taxonomy based on where and why erroneous behaviors occur as deviations from task models. To accomplish this, we defined our taxonomy using the Enhanced Operator Function Model (EOFM) (an example is shown in Fig. 1). EOFM is appropriate for this because it has a formal semantics (Fig. 2) that describes exactly how a task model rendered in it can be performed. This treats every activity and action as a finite state machine to explicitly represent how a task should execute normatively. A human who erroneously diverges from a task will do so by violating the formal semantics of a task. Thus, our taxonomy classifies erroneous human behavior based on how the formal semantics are violated. The nature of the violation shows us at which activity or action the divergent behavior occurred, what the erroneous behaviors was, and which part of the formal semantics were violated (not properly attended to by the human).

The taxonomy's classifications are hierarchical. First, it distinguishes between erroneous behaviors that originate at activity or action levels. Task execution can diverge from the formal semantics through violations of the execution state transition semantics or through incorrect action variable assignments that occur during action execution. Thus our taxonomy identifies the divergence type associated with the erroneous act. Divergences have limited ways they can manifest (erroneous behavior modes), the next taxonomy level. Within each mode, we classify an erroneous behavior based on the point of divergence from the semantics. For transition-based behaviors, this represents the transition that occurs (Fig. 3). For execution-based behaviors (Table 1) this is the type of variable assignment / action performed. Each point of divergence is refined to specific types based on what the human improperly attends to (for transition-based erroneous behaviors) or the improper variable assignment (for execution-based errors). Ultimately, we identified 52 erroneous behavior types (Fig. 4; see [3] for details).

Task 1.2: Taxonomy evaluation
To show that our new taxonomy encompasses both phenomenological and genotypical erroneous behaviors, we compared it to the leading models from both categories: Hollnagel's phenotypes [11] and Reason's slips [12]. Specifically, we demonstrated how all of the original taxonomies' behaviors were accounted for in our system (see [3]).

Goal 2

Task 2.1: Generation technique development
Existing EOFM-based erroneous behavior generation techniques [6, 8] were extended to support the new taxonomy. The new method was incorporated into EOFM-supported model checking analyses [4, 7, 9] (see Fig. 5) in the Symbolic Analysis Laboratory [10, 7]. This allowed erroneous behaviors to be generated based on the erroneous behavior modes and types (Fig. 3 and Table 1). For each analysis, an analyst can specify the maximum number of erroneous behaviors. He or she can also specify where the erroneous behaviors are generated. Generation can be applied for all activities and actions or specific ones. An analyst can also indicate if all or specific erroneous behaviors modes are generated.

Task 2.2: Generation technique testing
As features were added to erroneous behavior generation, code inspection and model checking analyses were used to ensure that the desired behaviors were being generated as intended.

Testing also evaluated the generation approach's scalability. Analyses were run using the tasks in Fig. 6. These increases in complexity between tasks based on the number of actions, activities, and strategic knowledge conditions. Each task was translated into a formal model and paired with a system model that responded to human actions. In all tasks, the human had to perform a given action (i.e. Action1) until its associated condition (i.e. C1) was true.

The maximum number of erroneous human behaviors included (applied to all activities and actions) was varied from 0 to 16. A true specification was verified against each model and the number of visited states and verification times were recorded. The results (Fig. 7) show that both values scale linearly with the maximum number of erroneous human behaviors. This is significant because it shows that the generation method can be applied to complex systems without the risk of exacerbating scalability.

Task 2.3: Case studies
To demonstrate the ability of this new method to discover system failures, we have used it replicate older analyses

[6, 8]: a radiation therapy machine [8] (previously evaluated with generated erroneous phenotypes [11]) and a patient controlled analgesia pump [6] (previously evaluated with generated slips [12]). When the new method was applied (allowing for all possible erroneous behaviors), it found the problems from previous analyses. This is significant because the original erroneous behavior generation methods were incompatible and would not find the problems discovered by the other. This provides further evidence that the new taxonomy and generation method account for both the phenotypes of erroneous action [11] and Reason's slips [12].

We also applied the new method to Army-relevant case studies. These included an unmanned aerial system (UAS) tasks for responding to low fuel conditions (Figs. 8 and 9) and an Apache helicopter where pilots fly between targets and identify whether to fire on them (Figs 10 – 14). Both applications have documented problems. The Apache helicopter was involved in incidents of fratricide [1] and the UAS fuel tasks have been associated with crashes [2].

For both applications, we used accident reports to model the human tasks [1, 2] and the automation the human interacted with. We evaluated each case with normative and generated erroneous human behavior. To allow for the discovery of multiple failures, we selectively generated all possible erroneous behaviors for each activity and action between evaluations. Thus, all possible erroneous behaviors that could manifest at a single task activity or action were considered in a given analysis.

The results (see Table 2) showed that there were 18 erroneous behaviors that could cause a UAS crash. One occurred when the human responded to a lack of fuel in the UAS's fuel tanks, but erroneously made an omission by completing the activity without shutting off the engine (an erroneous executing-to-done transition manifesting as an activity spurious termination omission for aRespondToFuelChange; see Table 2 and Fig. 8). This is the same condition that previously caused a crash [2]. We also found other failures. For example, humans may erroneously toggle fuel tank solenoids (various hToggleSolenoid done-to-executing and ready-to-executing intrusions; see Table 2 and Fig. 8). This can close the only tank with fuel or open empty ones, both conditions that can cause crashes.

For the Apache case, we found 39 erroneous behaviors that could lead to friendly fire (see Table 3). For example, a pilot could erroneously get permission to fire by reading the wrong value off of a coordinate display when getting radio clearance to fire (an Action Value Substitution for the hGetFireClearance from the new taxonomy; see Table 3 and Fig. 13). Similar conditions have been reported in fratricide incidents [1]. The analyses also revealed other failures. For example, the pilot could accidentally select and fly to the incorrect location (a Done-to-Executing intrusion classified as an Activity Capture, Reset Intrusion for aSelectDisplayLocation; see Table 3 and Fig. 10) and get clearance to fire based his or her comprehension of what the location should have been.

These analyses show that our method can find known and previously unknown problems in human-interactive systems. The identified problems should enable the Army to take measures to prevent them from occurring. A full accounting of the case studies will appear in a forthcoming journal article.

Training Opportunities: This project has supported two MS/Ph.D. students.

Kylie Molinaro was supported on the grant in the Fall 2015, Spring 2016, and Fall 2016 semesters.

Adam Houser was supported on the grant in the Spring 2017 and Summer 2017 semesters.

These students helped identify, model, and analyze the case studies used in the presented results. They also provided support for the PI during the development of the taxonomy.

Kylie Molinaro's MS was conferred in May 2016.

Results Dissemination: This project has produced one journal article discussing the erroneous behavior taxonomy:

Bolton, M. L. (2017). A task-based taxonomy of erroneous human behavior. The International Journal of Human-Computer Studies, 108, 105-121. https://doi.org/10.1016/j.ijhcs.2017.06.006

A second journal article documenting the erroneous behavior generation approach and the case study analyses is currently being written. Planned submission is Spring 2017.

RPPR Final Report
as of 17-Nov-2017

Honors and Awards: Nothing to Report

Protocol Activity Status:

Technology Transfer: The computer program for automatically generating erroneous human behaviors in task models will be freely available on the EOFM website: http://fhsl.eng.buffalo.edu/EOFM/

PARTICIPANTS:

Participant Type: PD/PI
Participant: Matthew Lee Bolton
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Graduate Student (research assistant)
Participant: Kylie Molinaro Molinaro
Person Months Worked: 13.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Graduate Student (research assistant)
Participant: Adam Houser
Person Months Worked: 7.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Bibliography

[1] Operation Desert Storm - Apache Helicopter Fratricide Incident: Report to the Chairman, Subcommittee on Oversight and Investigations, Conmittee on Energy and Commerce, House of Representatives. Technical Report GAO/OSI-93-4, 1993.

[2] Safety accident investigation report of US Army class A accident: A160T Hummingbird UAS #007. Technical Report 2010-07-28-1 322-XX-00007, Fort Belvoir, 2010.

[3] A task-based taxonomy of erroneous human behavior. *International Journal of Human-Computer Studies*, 108:105–121, 2017.

[4] M. L. Bolton & E. J. Bass. Formally verifying human-automation interaction as part of a system model: Limitations and tradeoffs. *Innovations in Systems and Software Engineering: A NASA Journal*, 6(3):219–231, 2010.

[5] M. L. Bolton & E. J. Bass. Using task analytic models to visualize model checker counterexamples. In *Proceedings of the 2010 IEEE International Conference on Systems, Man, and Cybernetics*, 2069–2074, Piscataway, 2010. IEEE.

[6] M. L. Bolton & E. J. Bass. Generating erroneous human behavior from strategic knowledge in task models and evaluating its impact on system safety with model checking. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 43(6):1314–1327, 2013.

[7] M. L. Bolton, R. I. Siminiceanu, & E. J. Bass. A systematic approach to model checking human-automation interaction using task-analytic models. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 41(5): 961–976, 2011.

[8] M. L. Bolton, E. J. Bass, & R. I. Siminiceanu. Generating phenotypical erroneous human behavior to evaluate humanautomation interaction using model checking. *International Journal of Human-Computer Studies*, 70 (11):888–906, 2012.

[9] M. L. Bolton, N. Jimenez, M. M. van Paassen, & M. Trujillo. Automatically generating specification properties from task models for the formal verification of human-automation interaction. *IEEE Transactions on Human-Machine Systems*, 44:561–575, 2014.

[10] L. De Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, & A. Tiwari. Sal 2. In *Computer aided verification*, 496–500. Springer, 2004.

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

[12] J. Reason. *Human Error*. Cambridge University Press, New York, 1990. ISBN 0521314194.

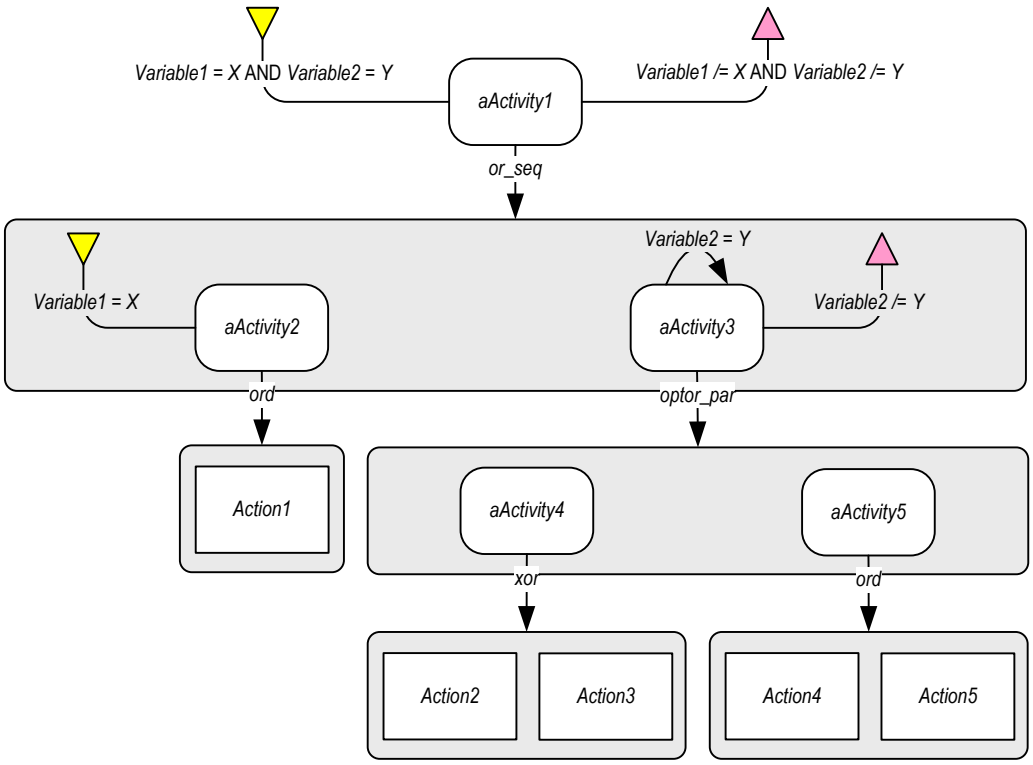


Figure 1: An example of a visualization of an EOFM task structure represented as a tree-like graph (note that a real EOFM will have more descriptive labels) [5]. Actions are rectangles and activities are rounded rectangles. An activity can decompose into other sub-activities and actions. A decomposition is presented as an arrow, labeled with the decomposition operator (which controls how sub-activities and actions execute in relation to each other), that points to a large rounded rectangle containing the decomposed activities or actions. Conditions (strategic knowledge) on activities are represented as shapes or arrows (annotated with the 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.

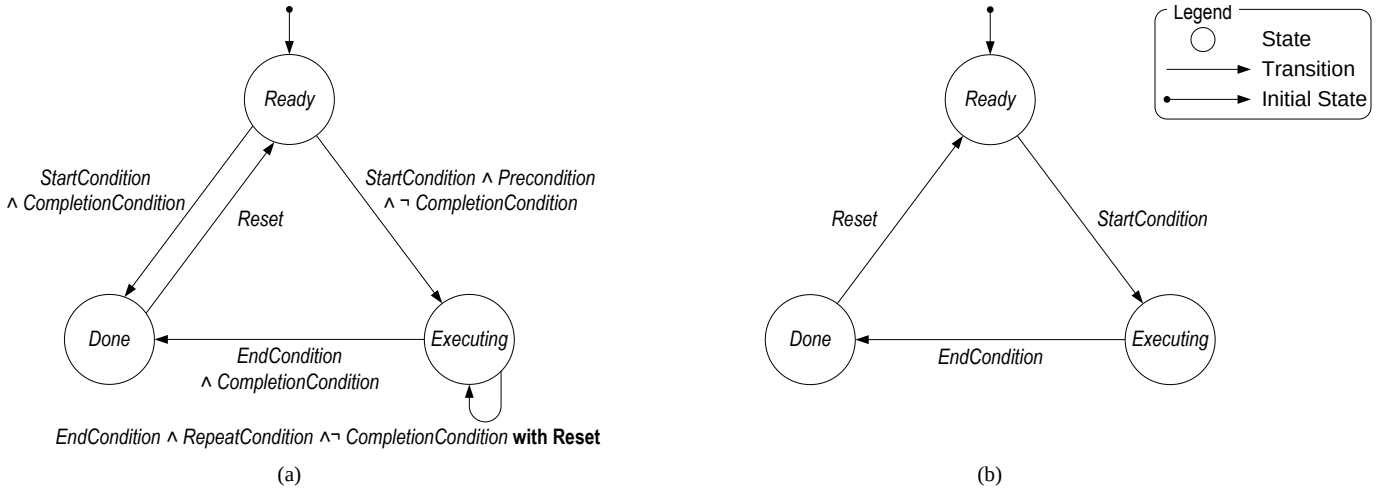


Figure 2: Formal semantics of an EOFM (a) activity's and (b) action's execution state presented as finite state transition systems [7]. Transitions are labeled with Boolean expressions that allow a transition to occur when they are true. *StarCondtion*s, *EndCondition*s, and *Reset*s express when an activity can start, end, or reset (respectively) based on the execution state of its parent, siblings (activities or actions in the same decomposition), and children (activities or actions it decomposes into). *Precondition*s, *RepeatCondtion*s, and *CompletionCondtition*s are explicitly specified by the modeling in the EOFM model and specify what must be true before an activity can execute, repeat, or complete (respectively). Note that **with Reset** is used to indicate a situation where all descendant activities and actions are *Reset* when the associated transition occurs.

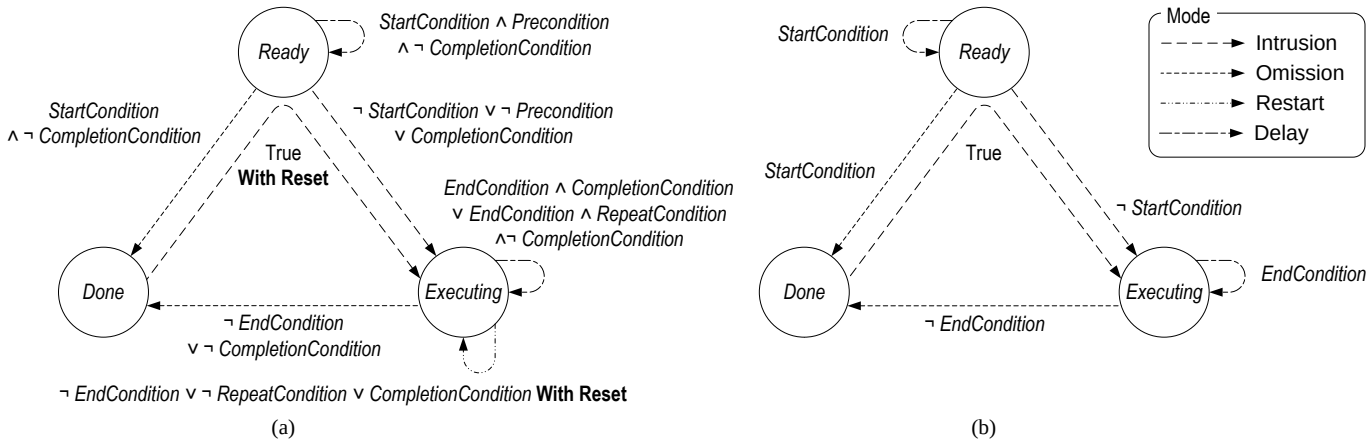


Figure 3: EOFM erroneous transition semantics.

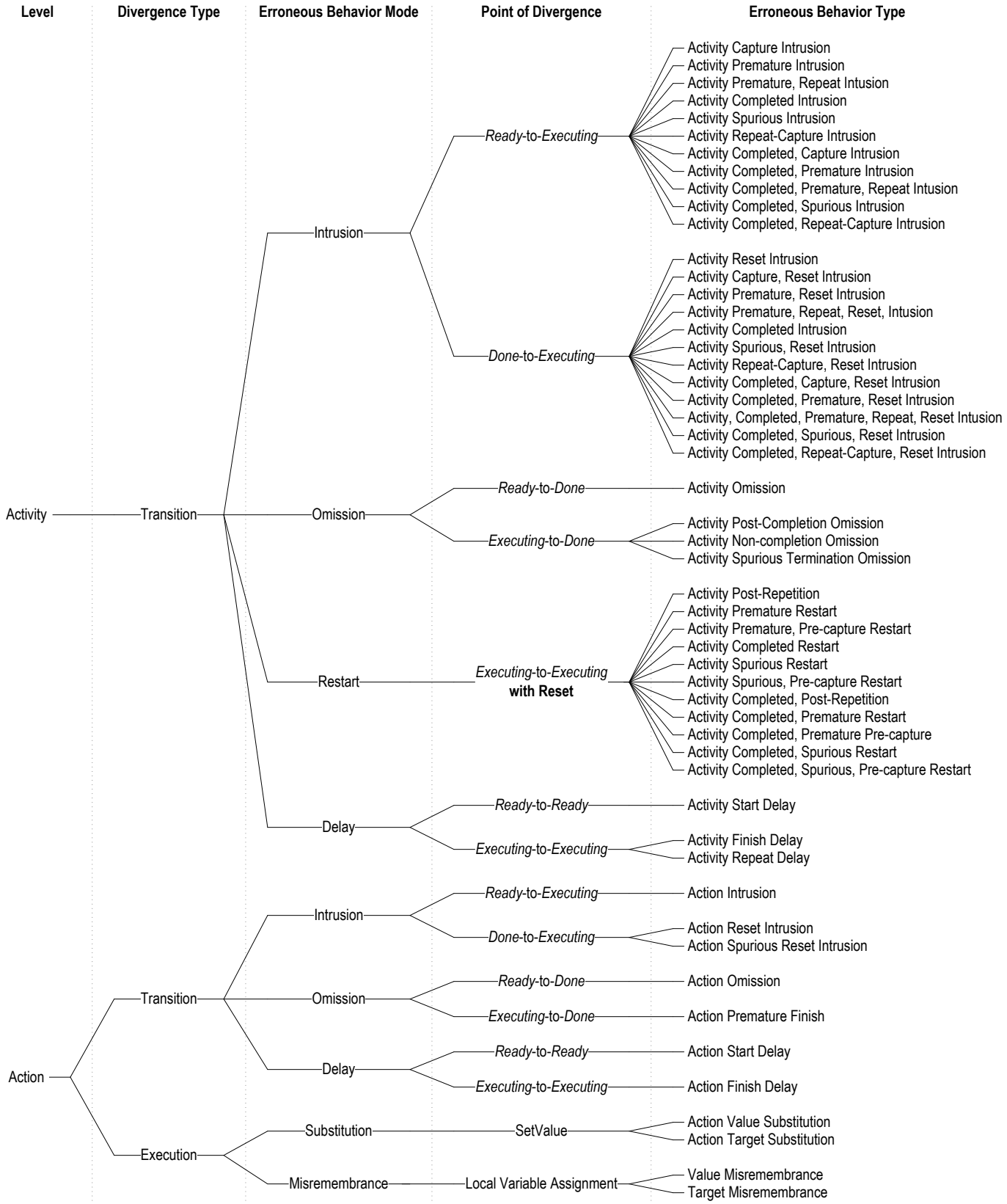


Figure 4: Summary of the classifications of the task-based taxonomy of erroneous human behavior.

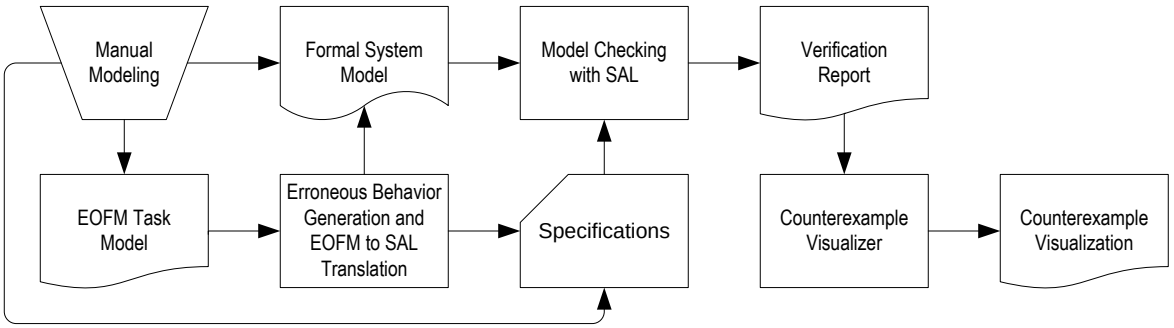


Figure 5: EOFM model checking analysis with the novel erroneous behavior generation technique incorporated into it. In this process, an analyst creates a normative task model as an EOFM. Within this model, the analyst can specify how to employ the erroneous human behavior generation technique: the maximum number of erroneous behaviors to include, whether erroneous behavior generation is applied to all activities and actions or specific ones and, if the latter, which erroneous behavior modes to include. The analyst also constructs a formal system model representing the behavior of the system and environment the human interacts with. The EOFM to SAL translator uses the EOFM formal semantics (Fig. 2) and the erroneous behavior semantics (Fig. 3 and Table 1) to automatically incorporate both the normative EOFM task behavior and generated erroneous behaviors into the formal system model. The SAL model checker will then be used to evaluate whether system specifications created by the analyst or automatically generated from the EOFM task models [9] will always be true. A verification report will indicate if the specifications hold and, if not, show how they were violated with counterexamples. Counterexamples can be inspected using the EOFM counterexample visualizer [5].

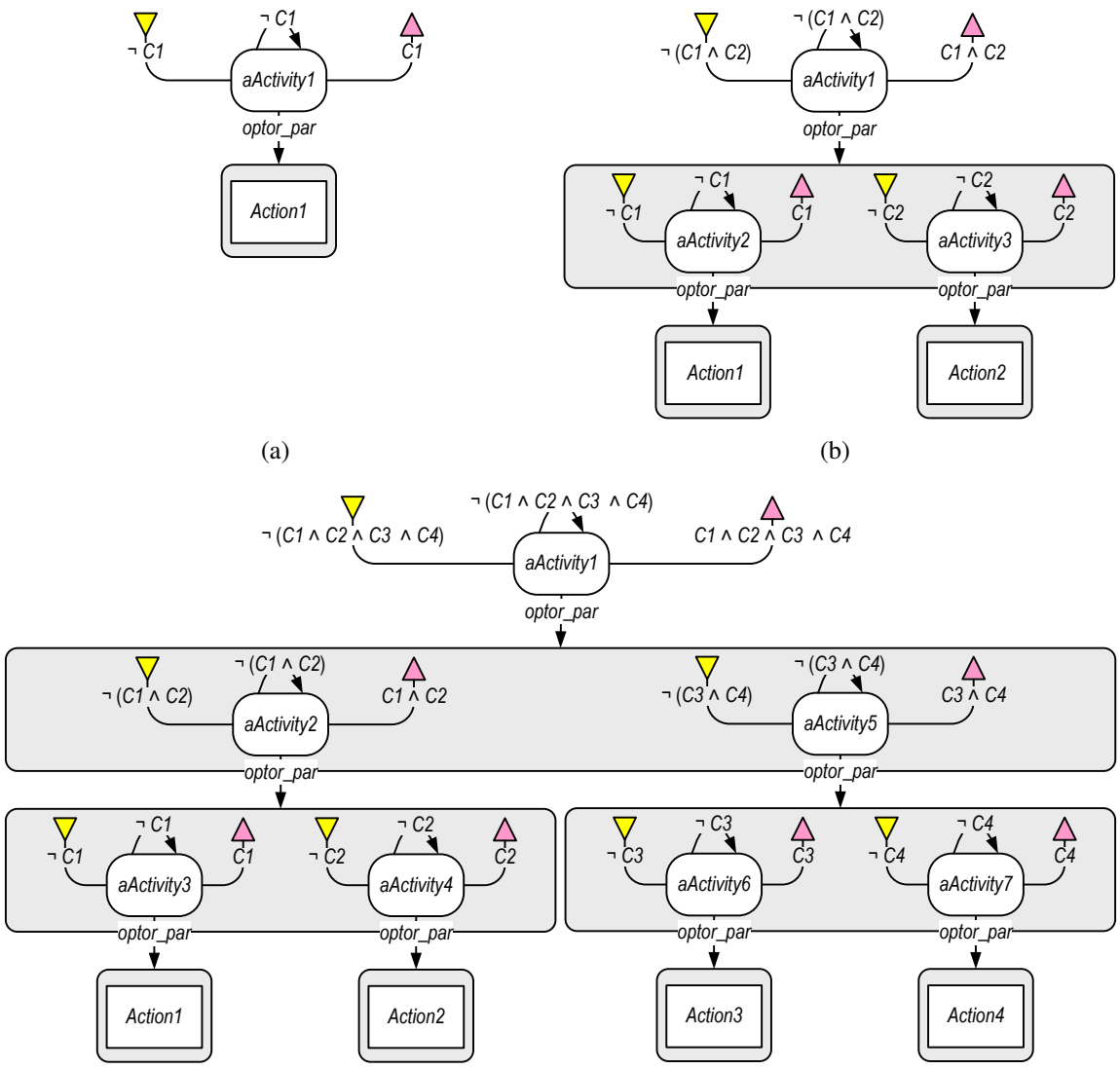


Figure 6: Instantiated EOFM normative task structures used as inputs to verification benchmark experiments. Activities begin with the letter "a" and actions do not. (a) The human operator must perform *Action1* until condition *C1* is true. (b) The human operator must perform *Action1* and *Action2* until *C1* and *C2* are true respectively. (c) The human operator must perform *Action1*, *Action2*, *Action3*, and *Action4* until *C1*, *C2*, *C3* and *C4* are true respectively.

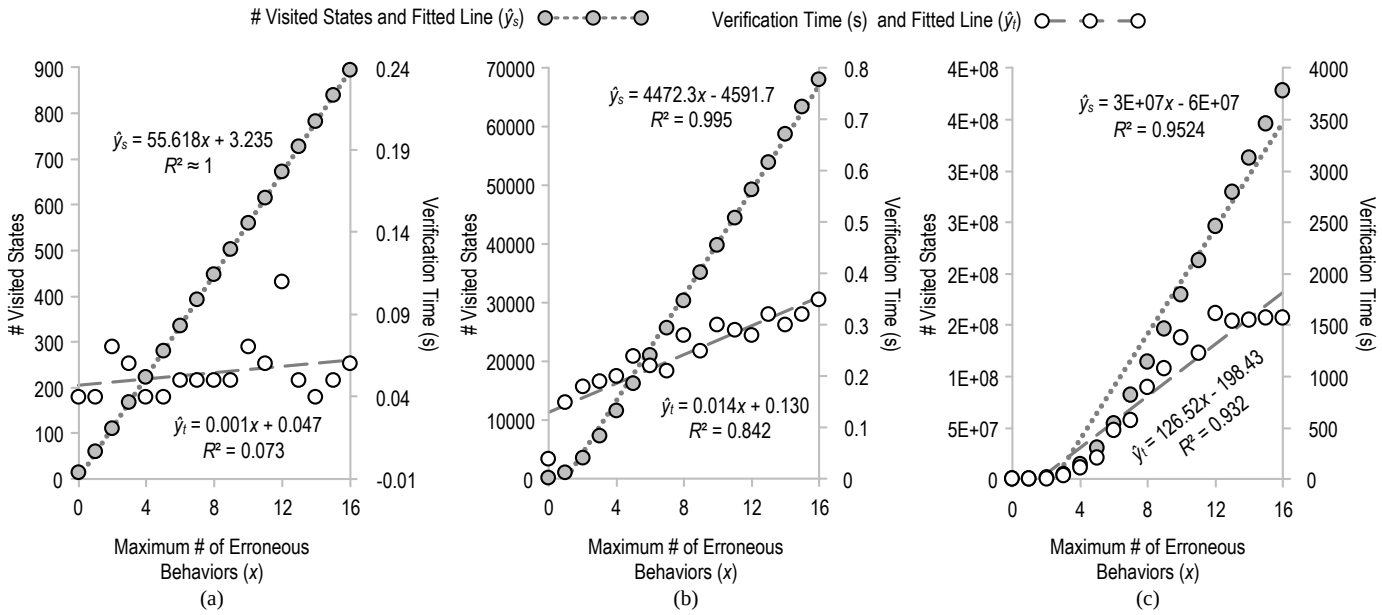


Figure 7: Scalability test results. These show that formal verification scales linearly (for both number of states and verification time) with the maximum number of erroneous behaviors included in the analyses.

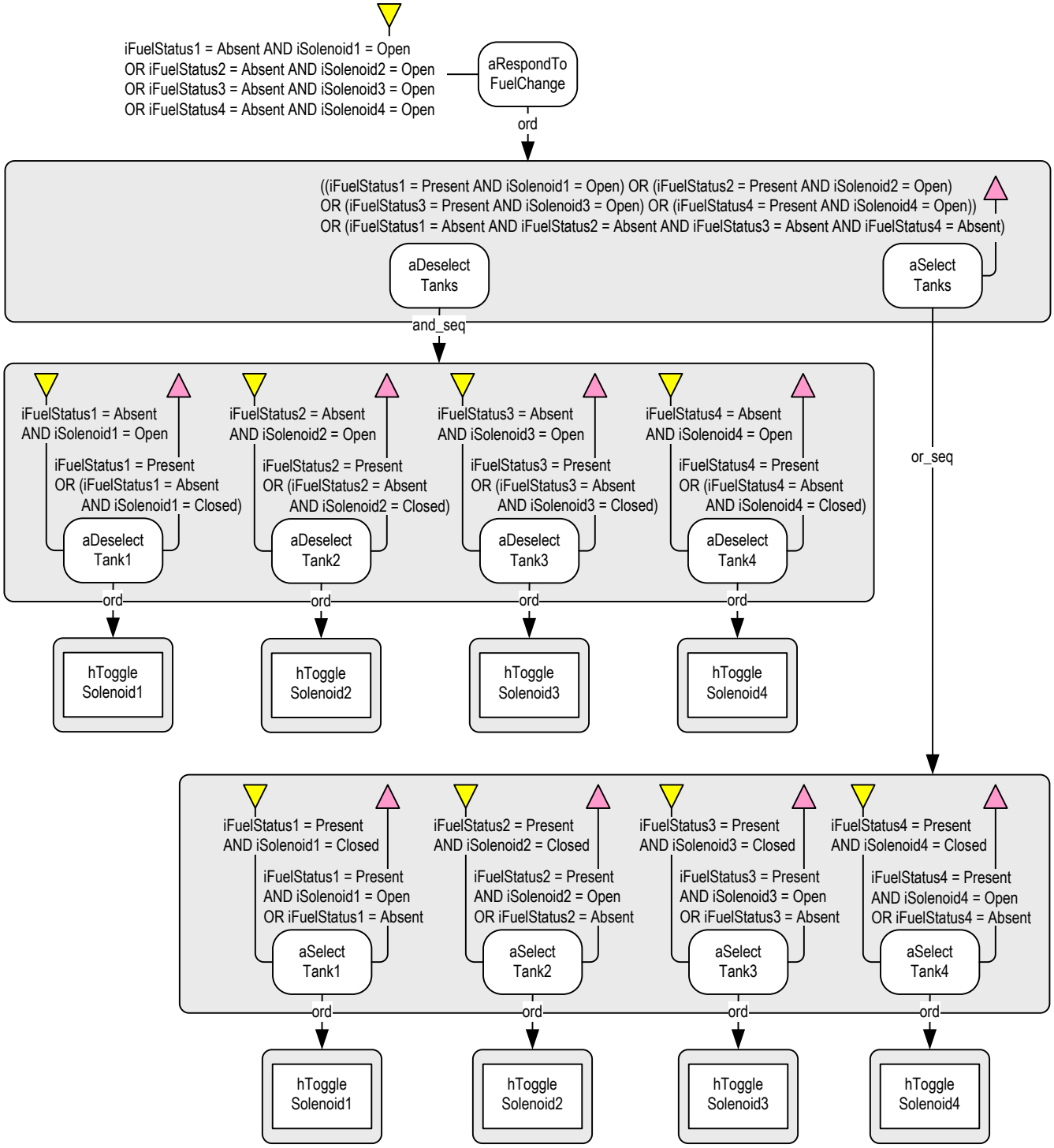
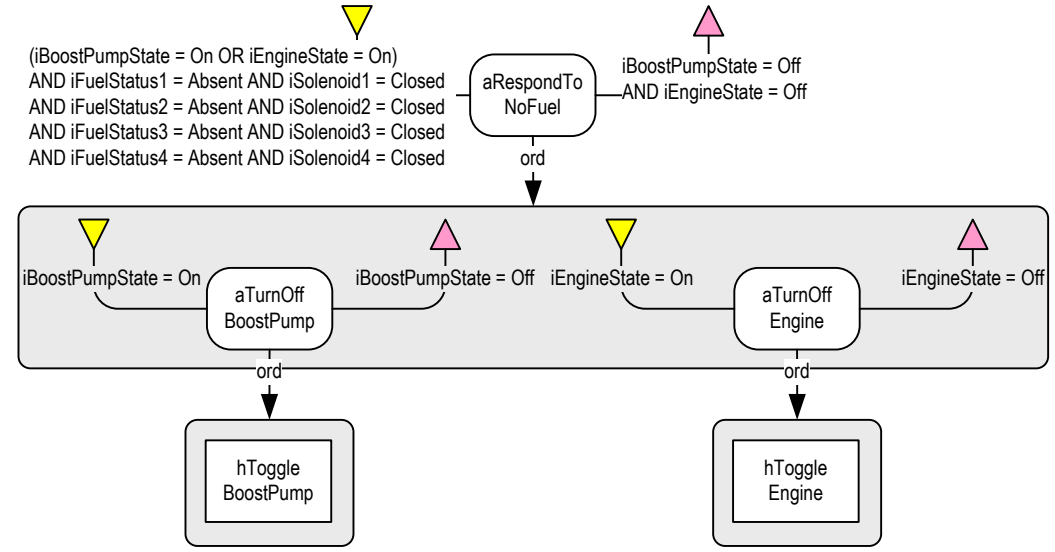
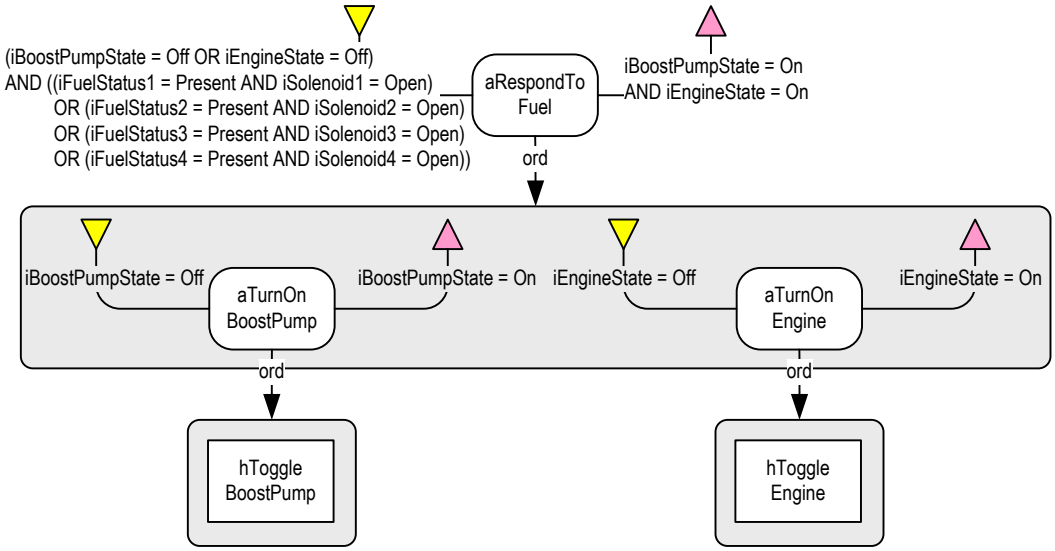


Figure 8: UAS application task for responding to a change in the UAS's fuel state. In this task the human can observe the status of the fuel in each of the UAS's four tanks, respectively associated with each of its four engines. The human can also observe the the state of four solenoids that can open or close the valve of a fuel tank to control whether fuel goes to the respective engine. In this situation, the human responds to a change in fuel state (a fuel tank becoming empty with its respective solenoid open). The human operator performs the task by first deselecting any tanks that do not have fuel by closing their solenoids. The pilot then has the ability to select any tanks (by opening their solenoids) that have fuel.



(a)



(b)

Figure 9: UAS application tasks for responding to a situation where (a) the UAS has no fuel and (b) the UAS has fuel. In (a), the human must toggle the UAS's boost pump and engine (in order) to turn them off. Conversely in (b), the human must toggle the UAS's boost pump and engine (in order) to turn them on.

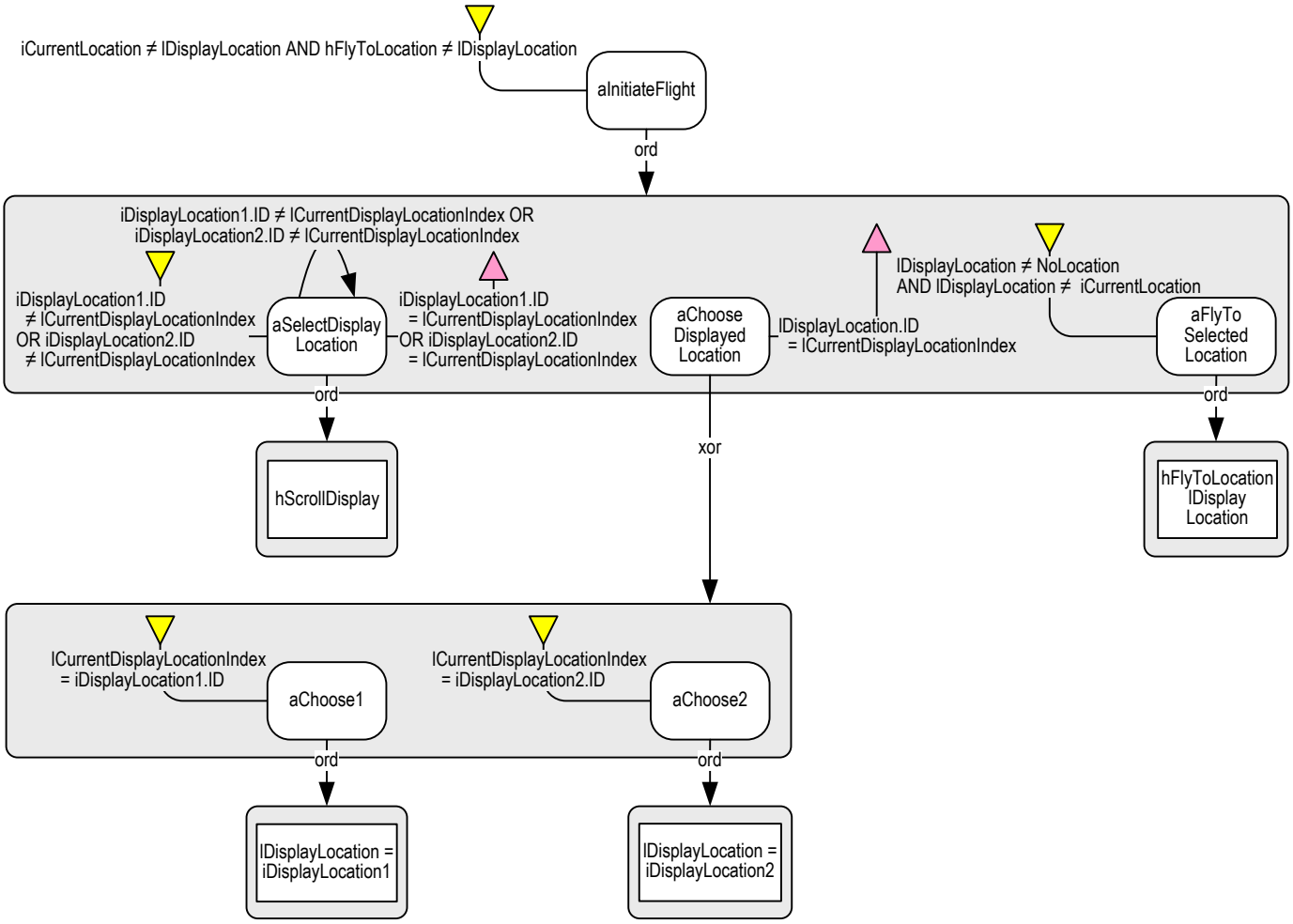


Figure 10: Task for initiating flight in the Apache helicopter application. This task can be performed if the helicopter's current location is not the displayed location in the computer and the helicopter is not enroute to that location. The pilot first scrolls through a display of locations in the helicopter's computer and notes the proper location from one of the two displayed on a given screen. He or she then sets the helicopter to fly to the noted location.

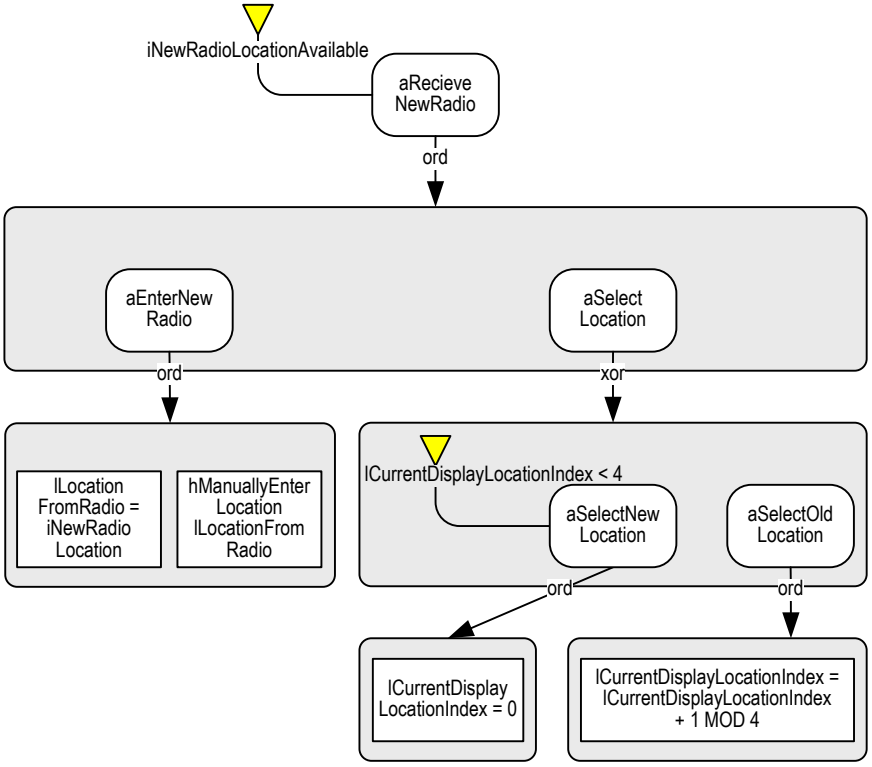


Figure 11: Task from the Apache application showing how a human pilot receives a new location over the radio, manually enters the location into the computer (which was inserted immediately after the current location), and then selects this location.

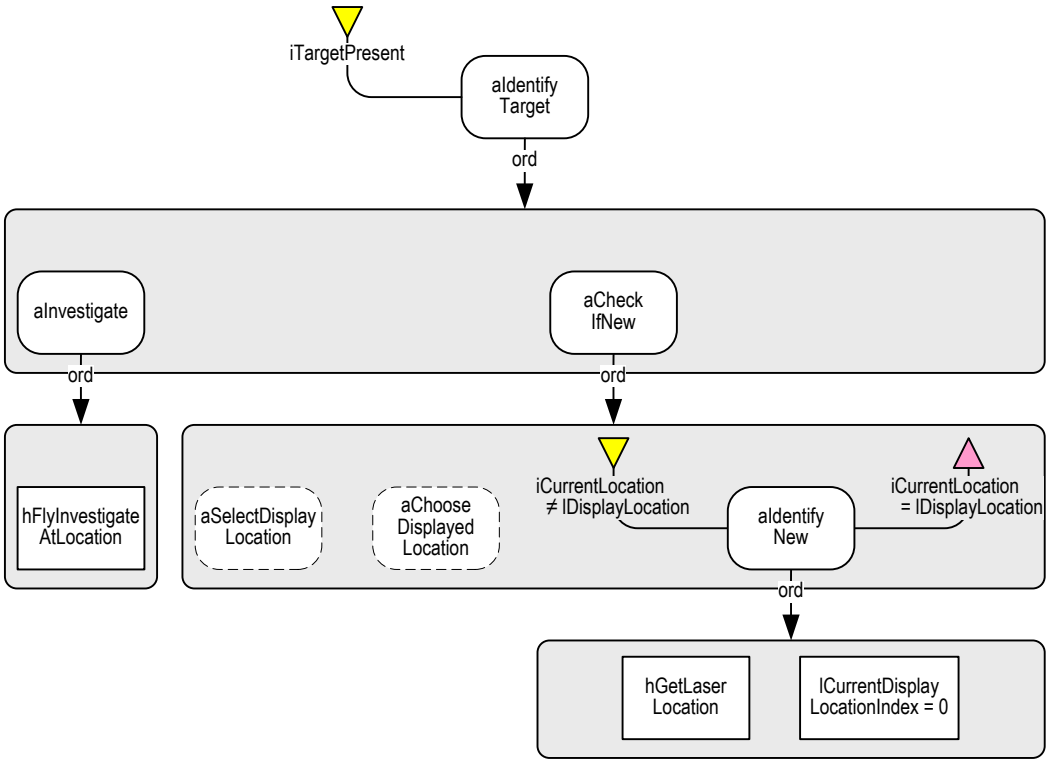


Figure 12: Task from the Apache application showing how a human pilot responds to the identification of a target in the environment. To do this, the pilot first hovers the aircraft by investigating at the location. The pilot then identifies if the target is new based on the target locations in the computer. If the target is new, the pilot uses the helicopter's laser to obtain the target's location. Note that in the above activities with dotted lines are define elsewhere (in this case Figure 9).

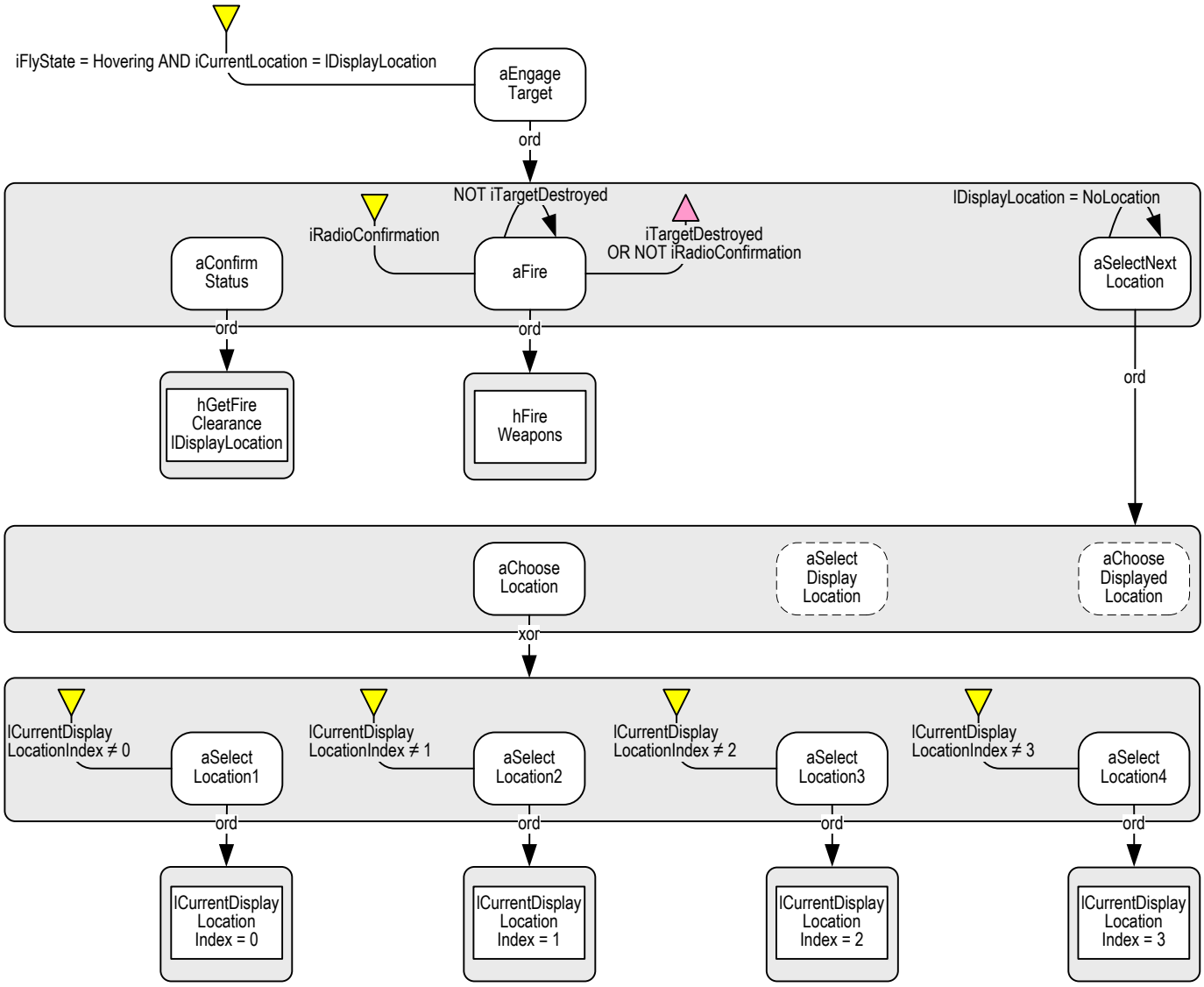


Figure 13: Task from the Apache application showing how a human pilot engages a target. If the helicopter is hovering and the helicopter is at the selected location, he or she can engage the helicopter. The pilot does this by first confirming whether or not to fire on the target by reading the displayed location over the radio. If a confirmation was received, the pilot can fire on the target until it is destroyed. The pilot must then find and select a new target to fly to.

Table 1: Action-level Erroneous Behaviors

Mode	Assignment	Erroneous Behavior Type
Substitution	*CorrectAction* := *IncorrectValue*	Action Value Substitution
	IncorrectAction := *CorrectValue*	Action Target Substitution
Misremembrance	*CorrectVariable* := *IncorrectValue*	Value Misremembrance
	IncorrectVariable := *CorrectValue*	Target Misremembrance

Table 2: Discovered UAS Application Errors and Verification Statistics.

Erroneous Activity / Action	Time (s)	# Visited states	Point of Divergence
aRespondToFuelChange	1.36	20452	ExecutingToDone
aDeselectTanks	1.36	85715	ReadyToDone
aDeselectTank1	1.60	215703	ReadyToDone
hToggleSolenoid1	2.32	551904	DoneToExecuting
aDeselectTank2	1.45	177366	ReadyToDone
hToggleSolenoid2	2.02	466413	ReadyToExecuting
aDeselectTank3	1.41	177366	ReadyToDone
hToggleSolenoid3	1.94	466413	ReadyToExecuting
aDeselectTank4	1.72	177366	ReadyToDone
hToggleSolenoid4	2.02	466413	ReadyToExecuting
aSelectTank1	2.66	743745	ReadyToExecuting
hToggleSolenoid1	2.25	604088	ReadyToExecuting
aSelectTank2	3.33	752466	ReadyToExecuting
hToggleSolenoid2	2.41	573977	ReadyToExecuting
aSelectTank3	3.05	752466	ReadyToExecuting
hToggleSolenoid3	2.31	573977	ReadyToExecuting
aSelectTank4	3.05	752466	ReadyToExecuting
hToggleSolenoid4	2.41	573977	ReadyToExecuting

Note. The Activity / Action map to actions and activities in Figs. 8 and 9.

Table 3: Discovered Apache Helicopter Application Errors and Verification Statistics.

Activity / Action	Verification Time (s)	# Visited states	Point of Divergence
aInitiateFlight	13.18	683140	ExecutingToDone
aSelectDisplayLocation	13.05	745977	DoneToExecuting
hScrollDisplay	10.87	409451	ReadyToExecuting
aChooseDisplayedLocation	11.55	508949	DoneToExecuting
aChoose1	11.50	478907	ReadyToExecuting
lDisplayLocation	9.90	1896915	LocalVariableAssignment_Misremembrance
aChoose2	10.28	378551	ReadyToExecuting
lDisplayLocation	9.54	1895321	LocalVariableAssignment_Misremembrance
aFlyToSelectedLocation	10.16	438422	DoneToExecuting
hFlyToLocation	9.81	527089	SetValue_ValueSubstitution
aRecieveNewRadio	12.98	909744	ReadyToExecuting
aEnterNewRadio	12.05	557998	ReadyToExecuting
hManuallyEnterLocation	11.51	1047300	SetValue_ValueSubstitution
aIdentifyTarget	13.26	864789	ExecutingToDone
aInvestigate	10.87	482260	DoneToExecuting
hFlyInvestigateAtLocation	11.14	348919	DoneToExecuting
aCheckIfNew	13.97	944480	ExecutingToDone
aSelectDisplayLocation	12.25	713170	DoneToExecuting
hScrollDisplay	10.89	418830	ReadyToExecuting
aChooseDisplayedLocation	11.62	443296	DoneToExecuting
aChoose1	11.35	466720	ReadyToExecuting
lDisplayLocation	10.24	1945480	LocalVariableAssignment_Misremembrance
aChoose2	10.35	427537	ReadyToExecuting
lDisplayLocation	9.21	1949899	LocalVariableAssignment_Misremembrance
aIdentifyNew	11.39	599971	DoneToExecuting
hGetLaserLocation	11.49	469858	ReadyToExecuting
aEngageTarget	6.83	31797	ReadyToExecuting
aConfirmStatus	9.60	270931	ReadyToDone
hGetFireClearance	8.91	198275	SetValue_ValueSubstitution
aFire	6.44	12642	ReadyToExecuting
hFireWeapons	6.53	10107	ReadyToExecuting
aSelectNextLocation	88.22	16301277	ReadyToExecuting
aSelectDisplayLocation	14.22	964988	ReadyToExecuting
hScrollDisplay	10.79	446885	ReadyToExecuting
aChooseDisplayedLocation	11.49	496445	ReadyToExecuting
aChoose1	11.04	484291	ReadyToExecuting
lDisplayLocation_1	11.04	1950732	LocalVariableAssignment_Misremembrance
aChoose2	10.40	485383	ReadyToExecuting
lDisplayLocation_2	10.63	1957855	LocalVariableAssignment_Misremembrance

Note. The Activity / Action map to actions and activities in Figs. 10–13.