2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 1-4, 2023, Oahu, Hawaii, USA

Negative Transfer in Task-based Human Reliability Analysis: A Formal Methods Approach*

Matthew L. Bolton[1], Svetlana Riabova[2], Yeonbin Son[1], and Eunsuk Kang[3]

Abstract— **Previous research has shown how statistical model checking can be used with human task behavior modeling and human reliability analysis to make realistic predictions about human errors and error rates. However, these efforts have not accounted for the impact that design changes can have on human reliability. In this research, we address this deficiency by using similarity theory from human cognitive modeling. This replicates how negative transfer can cause people to perform old task behaviors on modified systems. We present details about how this approach was realized with the PRISM model checker and the enhanced operator function model. We report results of a validation exercise using an application from the literature. We discuss the implications of our results and describe future research.**

I. Introduction

Human error is regularly cited as a source of system failure [1], [2], [3]. It has been a contributor to more than 1,000,000 injuries and between 44,000 and 98,000 deaths annually in medicine [4]; roughly 75% of all accidents in general aviation and 50% in commercial aviation [5], [6]; one third of unmanned aerial system (UAS) accidents [7]; 90% of automobile crashes [8]; and many high-profile disasters. Humans are often blamed for failures. However, the contemporary view presumes that human errors occur because of deeper system problems. Unfortunately, the complexity of modern systems and human-automation interaction (HAI) makes it extremely difficult to predict when and how human error can cause problems. For this reason, research has investigated how the proof techniques offered by formal methods can be used to evaluate and design HAI [9], [10], [11]. In particular, techniques have been developed where models of human tasks are coupled with models of system functionality. Then, formal verification/proof determines if normative or unexpected erroneous human behavior can result in safety violations [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23].

These powerful techniques are well-suited to discovering HAI design flaws. However, they have limitations. One is that errors and failures are traditionally modeled without probabilities. Thus, analysts must rely on intuition to determine which errors and/or failures are more risky and thus worthy of attention. Additionally, addressing errors is perilous because system changes can be disruptive and introduce new, previously unforeseen, errors [24]. For example, new tasks, even when adequately learned and practiced, may be more error-prone than tasks they replace [25], [1], [16], [26]. Worse, changes can create negative transfer [27], [28], a condition where a human's skilled behaviors encourage them to behave in ways that are now inappropriate [29].

In previous work, we introduced a method that combined human reliability analysis (HRA) with task-based erroneous behavior generation and formal verification to enable probabilistic predictions about human error and system failures [30]. In this paper, we extended this approach to account for negative transfer. In what follows, we present necessary background, the method we created, and a validation study based on an example from the literature. We ultimately discuss our results and suggest directions for future research.

II. Background

A. Formal Erroneous Human Behavior Generation

Task analysis is a systematic process that describes how humans normatively achieve goals with a system [31]. This is commonly documented as a hierarchical task model. Task models can be interpreted formally. This allows them to be included in a larger formal model that contains descriptions of other relevant system behaviors. Formal verification like model checking evaluates modeled behavior (including human errors generated in the task model) on system performance and safety (see [9] for a review). One of the most advanced formal task modeling languages (and the one used here) is the enhanced operator function model (EOFM).

EOFM [15], [32] is an XML-based task modeling formalism. EOFMs represent tasks as a hierarchy of goal-directed activities that ultimately decompose into actions. A decomposition operator specifies the temporal and cardinal relationships between decomposed acts: sequential or parallel execution, execution order, and how many can execute. Two of EOFM's nine decomposition operators are presented in this work. xor indicates that exactly one sub-act can execute. ord indicates that all must execute, one at a time, in their presented order. EOFMs also express task strategic knowledge explicitly as Boolean logical conditions on activities. These assert what must be true to start (preconditions), repeat (repeat conditions), and complete (completion conditions) execution.

Critically, EOFMs have formal semantics that enables inclusion in formal verification. For model checking, an automated translator [15] uses EOFM's formal semantics to convert a given task's XML into the input language of a model checker. This treats each activity and action as a state machine that transition between three states: ready, executing, and done. Transitions are based on Boolean conditions created using act strategic knowledge; the execution state of parent, sibling, and child acts; and the relationships between these dictated by task position and decomposition operators. This task state

*This material is based upon work supported by the National Science Foundation, USA under Grant Numbers 2219041, 1918314 and 1918140.

[1]Matthew L. Bolton and Yeonbin Son are with the University of Virginia, Charlottesvile, VA 22903, USA mlb4b@virginia.edu, ybson@virginia.edu

[2]Svetlana Riabova is with the University at Buffalo, Buffalo, NY 14260, USA sriabova@buffalo.edu

[3]Eunsuk Kang is with Carnegie Mellon, Pittsburgh, PA 15213, USA eunsukk@andrew.cmu.edu

2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 979-8-3503-3702-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/SMC53992.2023.10394333

TABLE I
EQ. (1) PARAMETERS FOR DIFFERENT COGNITIVE FUNCTIONS [34]

Parameter	Cognitive Function			
	Observation	Interpretation	Planning	Execution
a	0.0055	0.0041	0.0052	0.0065
b	-0.2458	-0.2046	-0.2828	-0.2860
c	0.2840	0.2244	0.4019	0.4079
d	-2.0775	-1.3495	-2.0000	-2.4120

machine model is composed with formal models of the rest of the system [14] for end-to-end behavioral verification.

When erroneous behavior generation is used with EOFM, a version of the formal model is created that allows task formal semantics to be violated in accordance with different theories of human error [33], [18], [19]. This allows model checking to discover how human errors could cause failures.

More recent work showed that these analyses could be merged with human error rate predictions based on a variant [34] of the Cognitive Reliability Error Analysis Method (CREAM) [35]. In this, each task (or task part) has an analyst-specified *CPCSum*. This numerical value (presumably derived from a subject-matter-expert) indicates how well the environment supports the human based on common performance conditions (CPCs): quality of the organization, work conditions, human-machine support, procedures, simultaneous goals, time availability, time of day, work experience, and team collaboration. Each of these conditions can be rated as improving (1), reducing (-1), or not affecting (0) human performance. The *CPCSum* is the aggregate of these ratings. Then, based on a regression model Bedford et al. [34] fitted to a large human error probability database, human error rates are predicted by:

$$P_{\text{HumanError}} = 10^{\left(a \cdot CPC_{Sum}^2 + b \cdot CPC_{Sum} + c + d\right)}. \quad (1)$$

Here, a, b, and c are determined by the cognitive function of the task and d is the $\log_{10}$ of the nominal error probability. Importantly, this makes predictions for four cognitive functions (Table I): observation, interpretation, planning, and execution. These are used in different parts of task execution and associated with different human errors [34], [35], [30].

When this was used with EOFM [30], our automatic translator would read in an EOFM and create a formal model in the input language of the PRISM model checker [36]. The formal model was based on the architecture in Fig. 1, where the human task model interacts with the formal representation of the other system elements. Importantly, the formal task took inputs from concurrent (synchronously composed) cognitive function models. These, using a *CPCSum* Eq. (1), probabilistically indicated if an error in each function occurred in each modeled step. The task model would see these inputs and, if the current part of task execution was associated with a given cognitive function, execute the task normatively or erroneously accordingly (see [30] for details).

This approach was able to accurately predict the probability of post completion errors for different versions of an ATM [30]. However, the method could not account for negative transfer that can occur during a design change.

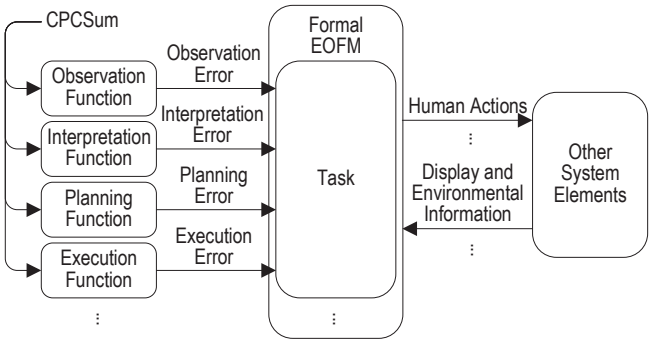

Fig. 1. Architecture used for supporting prediction of probabilities of human error with EOFM in formal methods analyses.

B. Negative Transfer and Similarity

Negative transfer occurs in a design change when the alterations are incompatible with the way the task was previously performed [27], [28]. This can result in the human performing an old (wrong) act. In particular, negative transfer occurs when two conditions are present. First there needs to be "surface similarity" between conditions (interface or environment states) that trigger changes in tasks. There also needs to be "structure discrepancy" (the tasks are performed differently) between the tasks that are triggered [27].

"Surface similarity" has been studied in cognitive science. In rule-based models of human cognition [37], humans recognize situations where different behaviors are appropriate based on whether features are present. This is typically expressed as a set of features being satisfied (in a Boolean antecedent expression where features are separated by "and"/$\wedge$ operators) that produce a result or consequent:

$$\text{IF } Feature_1 \wedge Feature_2 \wedge \ldots \wedge Feature_n \text{ THEN } Result. \quad (2)$$

Then, the similarity of two situations (A to B, with feature sets F_A and F_B respectively) is computed as the proportion of shared features to the total number of features of the situation being compared to (B) [38]:

$$Similarity = |F_A \cap F_B| / |F_B|. \quad (3)$$

III. OBJECTIVES

This research sought to use similarity from Eq. (3) [38] to account for negative transfer in our EOFM-based formal, probabilistic, error prediction. Specifically, in EOFM, surface similarity will manifest as similarity in the condition under which an activity or action will nominally execute. Furthermore, CREAM can account for problems with surface similarity by adjusting the *CPCSum* to account for the impact a change will have on error rates. Thus, we hypothesized that the impact of negative transfer could be incorporated in probabilistic EOFM error rate predictions by:

1) Explicitly including the replaced parts of the task in the formal representation of the task model;
2) Having the formal model (during execution) compute the surface similarity of the current situation to the original execution condition of replaced acts; and
3) Adjusting the CPCSum proportionally downwards based on the potential maximal impact on CPCs.

The following describes how we realized this approach in EOFM and PRISM.

TABLE II
INEQUALITY TO BOOLEAN EQUALITY EXPRESSION TRANSFORMATION

Original	Transformed
$X = x_i$	$(X = x_i)$
$X \neq x_i$	$(X = x_1) \quad \vee \ldots \vee (X = x_{i-1}) \vee (X = x_{i+1}) \vee \ldots \vee (X = x_n)$
$X < x_i$	$(X = x_1) \quad \vee \ldots \vee (X = x_{i-1})$
$X \leq x_i$	$(X = x_1) \quad \vee \ldots \vee (X = x_i)$
$X > x_i$	$(X = x_{i+1}) \vee \ldots \vee (X = x_i)$
$X \geq x_i$	$(X = x_i) \quad \vee \ldots \vee (X = x_i)$

Assume X is an integer variable with range $[x_1 \ldots x_n]$

IV. METHOD

A. Boolean Expression Parsing

The formal version of a task model provides the Boolean expression indicating when the original (old task) would execute. However, this can be expressed using combinations of equalities and inequalities across all the variables that describe a task or inputs to it. To be able to reason about these concepts with Boolean logic engines, these needed to be converted into simple equalities.[1] Fortunately, all variables in EOFM can be expressed purely as Boolean variables or finitely ranged integers. Thus, every part of such Boolean conditions can be converted to a Boolean expression of strict equalities as shown in Table II. In cases where equalities or inequalities express relationships between two variables, recursive algorithms iterate through possible values for each to create the aggregate ($\vee$) of all appropriate Boolean expressions.

Once these substitutions have been made, the Boolean expression can be converted into a form consistent with the antecedent of Eq. (2). This is accomplished by converting the formula into conjunctive normal form: a collection of Boolean "or" clauses ($\vee$) separated by "and" operations ($\wedge$). Thus, in this interpretation, each "or" clause represents a feature in surface similarity assessment.

If we let $\theta(Feature_i)$ be a function that evaluates a Boolean expression constituting a feature ($Feature_i$) that is 1 if the expression is true and 0 otherwise. Then, (in accordance with Eq. (3)) the degree of similarity a given situation has to an old context with n features is expressed by:

$$Similarity = \frac{\theta(Feature_1) + \ldots + \theta(Feature_n)}{n}. \quad (4)$$

B. Formal Model Integration

To account for negative transfer in the formal model, we extended our method to use the architecture in Fig. 2. When part of a task is replaced or modified, the original part is included in the formal EOFM representation. This Old Task Part is paired with three addition constructs:

1) SimilarityComputation is a formula (see Fig. 2) that used the original condition the task part would execute under as the condition from which features were extracted and similarity (Section IV-A) computed.
2) CPCAdjustment is a formula that uses the produced SimilarityScore to reduce the original CPCSum by an amount proportional to the Similarity Score:

$$\begin{aligned} \mathsf{AdjustedCPCSum} = &\mathsf{CPCSum} \\ &- \Delta \cdot \mathsf{SimilarityScore}, \end{aligned} \quad (5)$$

where Δ is the maximum CPC sum change. Because a change in a task relates to a human operator's knowledge about procedures, only the procedures CPC is adjusted. This restricts the range of Δ to [0, 2], where the actual upper bound would depend on the original assessment of the procedures CPC.

3) PlanningFunction' is a new concurrent module that uses the AdjustedCPCSum to compute whether a planning error occurs (PlanningError').

If PlanningError' indicates that a planning error is occurring, then the OldTaskPart is executed. To enable OldTaskPart to execute in place of the part of the replacing task, the formal model sets the execution state of the replacing task to done when OldTaskPart becomes done.

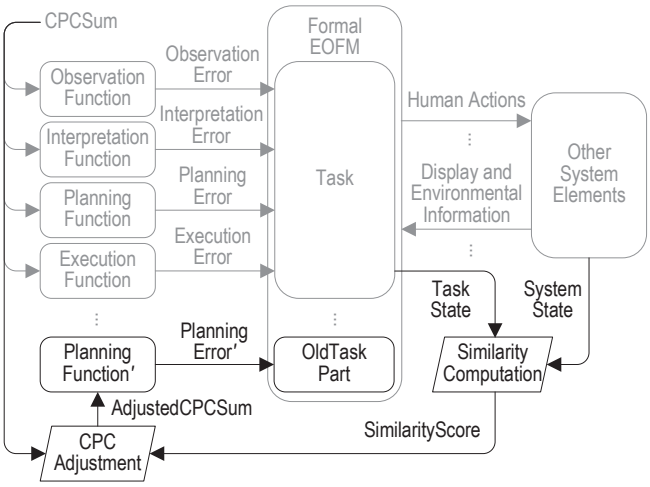

Fig. 2. Updated formal modeling architecture (from Fig. 1) for accounting for negative transfer. New elements use black lines and text. Parallelograms are formulas that compute values from other model variables during execution.

V. APPLICATION

As a first step towards validating our approach, we used it to model and evaluate a fruit packing case study introduced by Besnard et al. [29]. This was an appropriate application because Besnard et al. designed this problem to specifically evaluate the impact of negative transfer on human error.[2] In particular, the researchers designed a task where human users were expected to sort a selection of four fruit types (Bananas, Pears, Cherries, and Strawberries) into separate boxes and then (when a box was full with three fruits) packing the box into a larger container. In particular, participants used the interface in Fig. 3 to perform the following steps until all 9 of each types of fruits were processed:

1) **Point**: Instruct the system to randomly select a specific type of fruit to process.
2) **Select**: Select the box for the fruit that was pointed to. This is an iterative process where the user may need to perform multiple selects to cycle to the proper box.
3) **Fill / Empty**: If the box is not full (has less than three fruits), then the human adds one to the box (from the remaining unprocessed fruits) by performing the **Fill** operation. Otherwise, the human will perform the Empty operation to pack the three fruits into the larger container: leaving zero fruits in the box.

[1]We make use of the jbool_expressions library (https://github.com/bpodgursky/jbool_expressions).

[2]It also appears to the definitive source for empirical evidence of the effects of negative transfer.

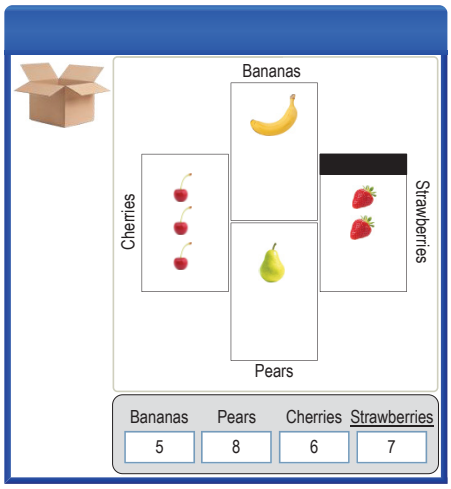

Fig. 3. Reproduction of the interface used for the fruit packing application [29]. The number of fruits remaining to be processed are shown at the bottom of the window. The pointed fruit is underlined. The fruit boxes are shown in the center of the display, where each box shows the number of each fruit in the box. The selected box has a black bar at its top. The large container in the upper left represents the container boxes are emptied into.

Originally, four separate keyboard keys were assigned to each action: Return ⏎ for point, → for select, Enter for fill, and Tab ⇄ for empty. However, after training people on this version of the task (and evaluating error rates), Besnard et al. [29] changed/swapped the keys' functions to investigate how negative transfer changed error rates: → for point, Enter for select, Tab ⇄ for fill, and Return ⏎ for empty.

A. Task Modeling

To apply our method to this domain, we first modeled the task in EOFM (see Fig. 4) to conform to the procedure described above. The original version of this model used the original key mapping employed by Besnard et al. [29].

B. Formal Modeling

We translated this model into PRISM's input language and paired this with a system behavior model. This kept track of the system state (corresponding to the pointing, selecting, and filling/emptying phases); the number of each fruit left to be sorted; the state of each fruit box; and pointing behavior.[3]

Because Besnard et al. observed low error rates in their original condition, we assumed a maximal CPCSum = 9. Then, to enable prediction about total numbers of errors, the formal model was augmented with a reward construct that would count the total number of erroneous actions: counting whenever the performed action did not match the state of the system (e.g., pointing when not in the pointing state).

C. Specification Properties

We formulated a specification property for calculating the expected number of errors to complete the task:

$$\mathbf{R}=?\left[\mathbf{F}\begin{pmatrix} iBananasLeft{=}0 & \wedge\ iPearsLeft{=}0 \\ \wedge\ iCherriesLeft{=}0 & \wedge\ iStrawberriesLeft{=}0 \\ \wedge\ sBananBox{=}0 & \wedge\ sPearBox{=}0 \\ \wedge\ sCherryBox{=}0 & \wedge\ sStrawberryBox{=}0 \end{pmatrix}\right]. \quad (6)$$

[3]A full listing of models used in this analysis can be found at https://drbolton.org/resources/

That is: calculate the expected reward/cost ($\mathbf{R}=?$; in this case the number of human errors) associated with eventually/finally ($\mathbf{F}$) reaching the system condition where there are no fruits left to be sorted and all the fruit boxes are empty.

D. Verification

The specification in Eq. (6) was verified against the formal model using PRISM's statistical model checker[4] with a path length of 10000 and 0.99 confidence. This was done on a laptop computer with an Intel Core i7-10510U CPU and 16 GB of RAM running Microsoft Windows 10. This showed that humans were expected to (on average) make 1.837 errors (with 99% confidence +/- 0.111 around this number) in the original, familiar condition. Verification took 23.771 seconds.

E. Model Modification

This normative model was then modified so that the swapped key configuration was used. In doing this (to account for negative transfer), each of the original model constructs for hitting the individual keys were treated as a separate OldTaskPart (Fig. 2) and given their own SimilarityComputation, CPCAdjustment, and PlanningFunction' constructs in accordance with the previous section. In all of these, the original CPCSum was modified based on similarity (in accordance with Eq. (5)) by using $\Delta = 2$. This was done because a CPCSum $= 9$ indicates all CPCs were assessed at the highest level, thus the procedures CPC would have a maximum Δ of 2 (going from improved to reduced). We felt that swapping keys was a significant enough change to constitute this shift.

F. Second Verification

With this new formal model for capturing the potential effect of negative transfer, we re-verified the specification from Eq. (6) (same analysis machine and settings). This time, verification took 214.574 seconds to show an expected 2.286 errors (99% confidence with +/- 0.121).

G. Results Comparison

We compared our results to those Besnard et al. [29] observed in an experiment with 10 human subjects.[5] This (Fig. 5) showed that our method accurately predicted the number of errors in the original condition: our prediction of 1.837 was between the experimentally observed range of [0,3] and was close to the observed average of 0.7. Additionally, our method made an accurate prediction of the impact of the change that negative transfer caused in the swapped condition: our prediction of 2.286 was in the experimentally observed range [2,15] and almost identical to the observed 2.3 average.

VI. Discussion

In this work, we introduced a new method to account for negative transfer in formal verification analyses that probabilistically predict human errors. This combines previous work that synthesized EOFM-based formal HAI and CREAM-based HRA with cognitive similarity theory. To provide

[4]Per the recommendations of Bolton et al. [30], statistical checking was used due to concerns about the scalability of probabilistic checking.

[5]Besnard et al. [29] ran three replications of each condition with the same participants. We are reporting the results from the first replication only because this best represents the negative transfer condition.

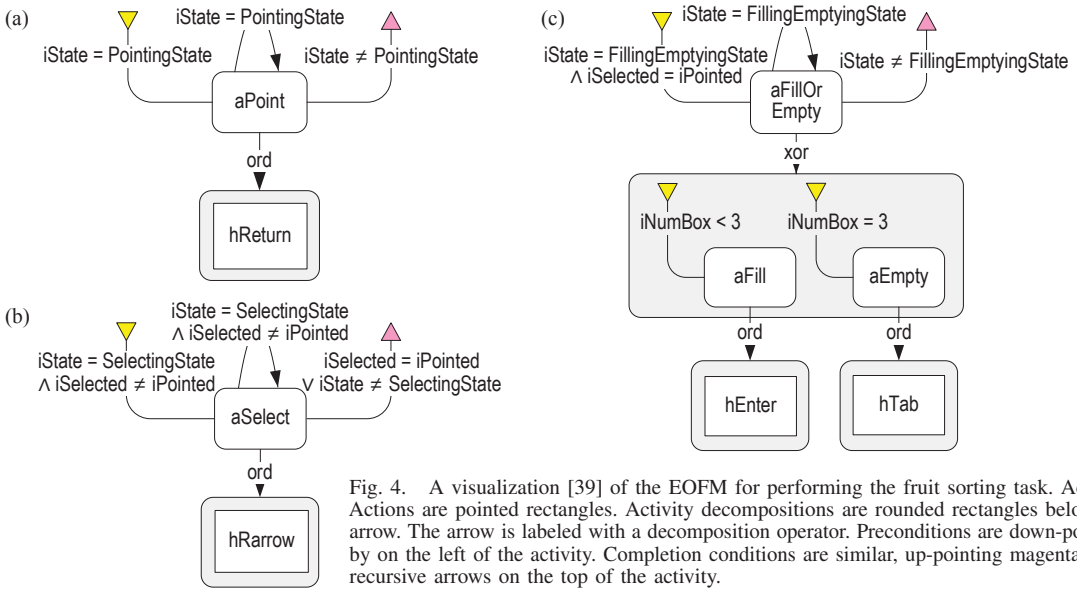

Fig. 4. A visualization [39] of the EOFM for performing the fruit sorting task. Activities are rounded rectangles. Actions are pointed rectangles. Activity decompositions are rounded rectangles below an activity, connected by an arrow. The arrow is labeled with a decomposition operator. Preconditions are down-pointing yellow arrows connected by on the left of the activity. Completion conditions are similar, up-pointing magenta triangle. Repeat conditions are recursive arrows on the top of the activity.

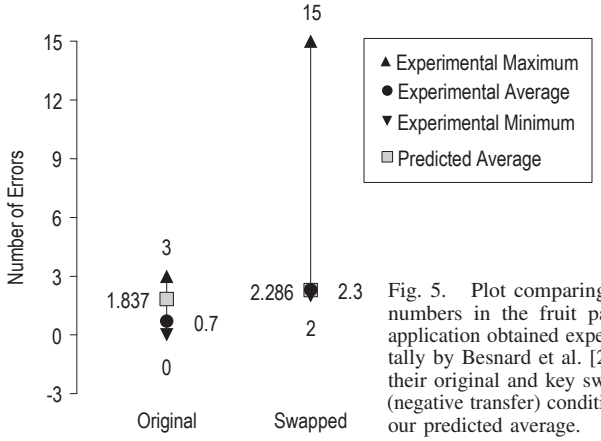

Fig. 5. Plot comparing error numbers in the fruit packing application obtained experimentally by Besnard et al. [29] for their original and key swapped (negative transfer) conditions to our predicted average.

a preliminary validation of this approach, we used it to replicate a leading experiment that demonstrated the effect of negative transfer on human error. That our method was able to accurately predict the number of errors both before and after the negative transfer condition, speaks to its validity. This is a significant contribution because it increases the accuracy of human error rate prediction and should allow analysts to evaluate the impacts of design changes across multiple design concepts. Because this is a model-based approach, these capabilities are achieved without having to evaluate all options with human subjects experiments. Additional contributions and validations of our method are expected in future work. This is discussed below.

A. EOFM Language and Translator Extensions

The implementation of our approach is currently semi-automated. The parts supported by previous work [30] are fully automated as is Boolean expression parsing and similarity formula creation (see Section IV-A and Eq. (4)). However, the other process steps are done manually. Current efforts are focusing on fully automating the model construction around the new architecture (Fig. 2) as an optional feature of the EOFM-to-PRISM translator. As part of this, language features are being added to the EOFM-XML notation to allow analysts to specify how parts of tasks were replaced in revisions.

B. Scalability

In the presented application, the original system verified in 23.771 seconds. The swapped keys condition (with negative transfer prediction) verified in 214.574 seconds: a 802.671% increase. This is a significant jump, possibly indicating limits on the scalability of the method.

Based on the combinatorics of formal models, the complexity increase is likely caused by the additional Planning Function' modules. This is because these are stateful and concurrent. As such, each increases total model state-space size by its number of states times the number of states in the remainder of the model. Our example application contained four such modules (one for each swapped action).

It may be the case that this feature will limit method applicability. However, four task changes would likely be an upper bound on typical application. We envision the standard use case for the method only making one task change in any given analysis. Thus, our method should still scale reasonably. In fact, if we create a version of the fruit packing model that only includes one additional Planning Function' module, this verifies in 29.835 seconds: a reasonable 25.510% increase.

Future efforts will more deeply explore method scalability and methods for improving it.

C. Additional Application and Validation

The fruit packing application [29] is admittedly simple. Given its prominence in the literature, it is an appropriate one for validating our approach. In fact, this was the only published study we could find that explicitly dealt with error rates and negative transfer. However, the small sample size and artificiality of the task leave room for improvement. In future work, we plan to model a real-world application in the form of the Open EMR electronic medical record system. We also plan to validate our prediction using a human subjects experiment involving this application.

D. Automated Model Repair

The effort documented here is a part of a larger project for quantifying human-machine reliability and automatically repairing interfaces [40], [30], [41], [42]. Future effort will investigate how negative transfer prediction can be automatically inform automated interface repair methods.

VII. Conclusions

This work demonstrated that probabilistic predictions about human error rates using task models, HRA, and statistical model checking can be made to account for negative transfer effects. This had important implications for engineering and making changes to safety-critical HAI. Future work will seek to validate results will a more realistic application and a more robust number of measured samples.

References

[1] J. Reason, *Human Error*. New York: Cambridge University Press, 1990.

[2] E. Hollnagel, "The phenotype of erroneous actions," *International Journal of Man-Machine Studies*, vol. 39, no. 1, pp. 1–32, 1993.

[3] T. B. Sheridan and R. Parasuraman, "Human-automation interaction," *Reviews of human factors and ergonomics*, vol. 1, no. 1, pp. 89–129, 2005.

[4] L. T. Kohn, J. Corrigan, and M. S. Donaldson, *To Err is Human: Building a Safer Health System*. Washington: National Academy Press, 2000.

[5] "The 32nd Joseph T. Nall report," 2023, https://www.aopa.org/training-and-safety/air-safety-institute/accident-analysis/joseph-t-nall-report.

[6] R. Kebabjian, "Accident statistics," 2023, planecrashinfo.com.

[7] S. D. Manning, C. E. Rash, P. A. LeDuc, R. K. Noback, and J. McKeon, "The role of human causal factors in US Army unmanned aerial vehicle accidents," USA Army Research Laboratory, Tech. Rep. 2004-11, 2004.

[8] NHTSA, "National motor vehicle crash causation survey: Report to congress," National Highway Traffic Safety Administration, Springfield, Tech. Rep. DOT HS 811 059, 2008.

[9] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Using formal verification to evaluate human-automation interaction in safety critical systems, a review." *IEEE Transactions on Systems, Man and Cybernetics: Systems*, vol. 43, no. 3, pp. 488–503, 2013.

[10] M. L. Bolton, "Novel developments in formal methods for human factors engineering," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. SAGE Publications Sage CA: Los Angeles, CA, 2017, pp. 715–717.

[11] B. Weyers, J. Bowen, A. Dix, and P. Palanque, Eds., *The Handbook of Formal Methods in Human-Computer Interaction*. Berlin: Springer, 2017.

[12] F. Paternò and C. Santoro, "Integrating model checking and HCI tools to help designers verify user interface properties," in *Proceedings of the 7th International Workshop on the Design, Specification, and Verification of Interactive Systems*. Berlin: Springer, 2001, pp. 135–150.

[13] Y. Aït-Ameur and M. Baron, "Formal and experimental validation approaches in HCI systems design based on a shared event B model," *International Journal on Software Tools for Technology Transfer*, vol. 8, no. 6, pp. 547–563, 2006.

[14] M. L. Bolton and 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*, vol. 6, no. 3, pp. 219–231, 2010.

[15] M. L. Bolton, R. I. Siminiceanu, and 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*, vol. 41, no. 5, pp. 961–976, 2011.

[16] R. Bastide and S. Basnyat, "Error patterns: Systematic investigation of deviations in task models," in *Task Models and Diagrams for Users Interface Design*. Berlin: Springer, 2007, pp. 109–121.

[17] R. E. Fields, "Analysis of erroneous actions in the design of critical systems," Ph.D. dissertation, University of York, York, 2001.

[18] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Generating phenotypical erroneous human behavior to evaluate human-automation interaction using model checking," *International Journal of Human-Computer Studies*, vol. 70, no. 11, pp. 888–906, 2012.

[19] M. L. Bolton and 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*, vol. 43, no. 6, pp. 1314–1327, 2013.

[20] ——, "Formal modeling of erroneous human behavior and its implications for model checking," in *Proceedings of the Sixth NASA Langley Formal Methods Workshop*. Hampton: NASA Langley Research Center, 2008, pp. 62–64.

[21] D. Pan and M. L. Bolton, "Properties for formally assessing the performance level of human-human collaborative procedures with miscommunications and erroneous human behavior," *International Journal of Industrial Ergonomics*, vol. 63, pp. 75–88, 2018.

[22] M. L. Bolton, "Model checking human–human communication protocols using task models and miscommunication generation," *Journal of Aerospace Information Systems*, vol. 12, no. 7, pp. 476–489, 2015.

[23] A. Barbosa, A. C. Paiva, and J. C. Campos, "Test case generation from mutated task models," in *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*. ACM, 2011, pp. 175–184.

[24] L. Bainbridge, "Ironies of automation," *Automatica*, vol. 19, no. 6, pp. 775–780, 1983.

[25] M. L. Bolton, "A task-based taxonomy of erroneous human behavior," *International Journal of Human-Computer Studies*, vol. 108, pp. 105–121, 2017.

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

[27] P. Johnson, "Supporting system design by analyzing current task knowledge," in *Task analysis for human-computer interaction*, D. Diaper, Ed. Ellis-Horwood, 1989, pp. 160–185.

[28] D. N. Perkins and G. Salomon, "Transfer of learning," *International encyclopedia of education*, vol. 2, pp. 6452–6457, 1992.

[29] D. Besnard and L. Cacitti, "Interface changes causing accidents. an empirical study of negative transfer," *International Journal of Human-Computer Studies*, vol. 62, no. 1, pp. 105–125, 2005.

[30] M. L. Bolton, X. Zheng, and E. Kang, "A formal method for including the probability of erroneous human task behavior in system analyses," *Reliability Engineering & System Safety*, vol. 213, p. 107764, 2021.

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

[32] M. L. Bolton and E. J. Bass, "Enhanced operator function model (EOFM): A task analytic modeling formalism for including human behavior in the verification of complex systems," in *The Handbook of Formal Methods in Human-Computer Interaction*, B. Weyers, J. Bowen, A. Dix, and P. Palanque, Eds. Cham: Springer, 2017, pp. 343–377.

[33] M. L. Bolton, K. A. Molinaro, and A. M. Houser, "A formal method for assessing the impact of task-based erroneous human behavior on system safety," *Reliability Engineering & System Safety*, vol. 188, pp. 168–180, 2019.

[34] T. Bedford, C. Bayley, and M. Revie, "Screening, sensitivity, and uncertainty for the CREAM method of human reliability analysis," *Reliability Engineering & System Safety*, vol. 115, pp. 100–110, 2013.

[35] E. Hollnagel, *Cognitive Reliability and Error Analysis Method (CREAM)*. Oxford: Elsevier, 1998.

[36] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *International Conference on Computer Aided Verification*. Springer, 2011, pp. 585–591.

[37] A. Tversky, "Features of similarity," *Psychological review*, vol. 84, no. 4, p. 327, 1977.

[38] R. Sun, "Robust reasoning: Integrating rule-based and similarity-based reasoning," *Artificial Intelligence*, vol. 75, no. 2, pp. 241–295, 1995.

[39] M. L. Bolton and E. J. Bass, "Using task analytic models to visualize model checker counterexamples," in *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2010, pp. 2069–2074.

[40] X. Zheng, M. L. Bolton, and C. Daly, "Extended safphR (Systems Analysis for Formal Pharmaceutical Human Reliability): Two approaches based on extended cream and a comparative analysis," *Safety Science*, vol. 132, p. 18 pages, 2020.

[41] P. Wan and M. L. Bolton, "A taxonomy of forcing functions for addressing human errors in human-machine interaction," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2021, pp. 3134–3139.

[42] C. Zhang, T. Saluja, R. Meira-Góes, M. Bolton, D. Garlan, and E. Kang, "Robustification of behavioral designs against environmental deviations," in *Proceedings of the 45th International Conference on Software Engineering*, 2023, p. 12 pages.