Validation of a Formal Method for Human Error Rate Prediction With Negative Transfer

Yeonbin Son, *Graduate Student Member, IEEE*, Matthew L. Bolton, *Senior Member, IEEE*, Emma Crooks, Hannah Palmer, Eunsuk Kang, and Christopher Daly

***Abstract*—Human error is often associated with system failures. The complexity of human-automation interaction can make it difficult to anticipate what errors can occur and how they contribute to failures. Previous research has shown that task analytic behavior modeling with the enhanced operator function model and the cognitive reliability analysis method (CREAM) can be combined with statistical model checking to make predictions about human error rates, their stochastic impact on system failures, and the effect of negative transfer of design changes on these predictions. These efforts were successful, but the validation studies used artificial examples with limited data. Predictions also slightly overestimated error rates. This article addresses these deficiencies by conducting a validation study based on the prescription order entry interface of the OpenEMR electronic medical record. As part of this, we explored how prediction accuracy for the OpenEMR application changed based on the inclusion/exclusion of planning errors: errors based on people's ability to formulate task plans, which we hypothesized contributed to error rate overestimation. Results found that our method's predictions aligned with those observed in the experiment, especially when planning errors were excluded. Negative transfer conditions did not manifest significant differences in error rates experimentally or in model predictions. These results suggest that negative transfer's impact on human–computer interaction may be overstated in the literature. Finally, higher error rates were observed between the original OpenEMR prescription order entry interface compared to an alternative that we tested. We highly suggest that OpenEMR adopt the alternative.**

***Index Terms*—Human error, human reliability, model checking, negative transfer.**

Received 22 August 2024; revised 9 May 2025 and 11 July 2025; accepted 24 July 2025. Date of publication 25 August 2025; date of current version 23 October 2025. This work was supported by the National Science Foundation, USA under Grant 2219041, Grant 1918314, and Grant 1918140. This article was recommended by Associate Editor C. Lv. *(Corresponding author: Matthew L. Bolton.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the University of Virginia Institutional Review Board for the Social and Behavioral Sciences under Application No. 5064 and performed in line with the Declaration of Helsinki.

Yeonbin Son, Matthew L. Bolton, and Hannah Palmer are with the Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA 22903 USA (e-mail: mlb4b@virginia.edu).

Emma Crooks is with US Army Corps of Engineers, Buffalo, NY 14207 USA.

Eunsuk Kang is with Software and Societal Systems Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Christopher Daly is with Department of Pharmacy Practice, University at Buffalo, Buffalo, NY 14214 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/THMS.2025.3593085.

Digital Object Identifier 10.1109/THMS.2025.3593085

I. Introduction

HUMAN error is often cited as a major factor in complex system failure [1], [2], [3]. It contributes to approximately 50% of commercial aviation accidents and 75% in general aviation [4], [5]; at least a third of unmanned aerial system (UAS) accidents [6]; and between 44 000 and 98 000 deaths in medicine annually [7], [8]. These errors often occur because of systemic issues, where engineers do not account for humans in their designs or the impact that human errors can have. However, modern system complexity can make it extremely challenging for engineering to account for all the human-automation interactions (HAIs) and errors that need to be considered. Thus, one way researchers have attempted to address this is by exploring how formal methods (techniques for proving properties about system models) can be used to evaluate and engineer complex, human-interactive systems [9], [10], [11], [12], [13]. The research presented here specifically focuses on methods that combine models of human task behavior with models of the automation's behavior. When this is done, formal verification/proof determines if included behavior (inclusive of normative human behavior and unexpected human errors) can contribute to a violation of system safety [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28].

These techniques are particularly good at discovering when and how HAI design flaws can be problematic, but with limitations. For example, such analyses are traditionally performed with deterministic or non-deterministic models. This allows them to account for various behaviors, interactions, and performance options, but without considering probabilities. As a result, decision-makers must use their intuition and domain knowledge to determine if discovered failure sequences are worthy of intervention. Furthermore, attempting to fix design flaws or errors can be fraught. This is because any system change could introduce unforeseen errors or confuse expert human operators [29]. New tasks can be more error-prone than those replaced [1], [18], [30], [31]. Changes can also create negative transfer [32], [33], where a human's skill in the old task encourages erroneous behavior with the new design [34].

In previous work, we introduced a method that combined human reliability analysis (HRA) with task-based erroneous behavior generation and formal verification with probabilistic and statistical model checking: tools for automatically proving properties about stochastic systems (based on the model or traces through the model, respectively) [35]. When used together in our method, we can make probabilistic predictions about human error and system failures [36]. We extended this method to account for the effects of negative transfer (of learning) using cognitive similarity theory [37]. While these efforts were

successful, the applications used to validate findings were based on artificial examples with limited data. Additionally, the error rates predicted (while completely reasonable) tended to be higher than those in human data.

To address these deficiencies, this research conducted a human subjects experiment, based on a human–machine interface from the OpenEMR electronic medical record system, that sought to validate the error rate predictions of our method, both with and without negative transfer. As part of this, we also explored whether removing planning errors from model computations would improve error rate predictions. In what follows, we present the necessary background for understanding our research. This includes a thorough description of our method. This is followed by a more precise description of the research's objectives. We then describe the human subjects experiment and modeling effort that were used to evaluate our method. After presenting our results, we discuss them and use this as the basis for suggesting future research directions.

II. Background

There is a vast literature on human error as well as design and analysis methods for addressing it. Below, we focus on that which is critical to our research. This includes work on formal verification with task models and human error, as well as extensions that make probabilistic predictions about human error and the modeling of negative transfer.

A. Formal Verification With Task Behavior and Human Errors

Task analysis is a systematic process that describes how humans normatively achieve goals with a system [38]. 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 in this research) is the enhanced operator function model (EOFM).

EOFM [17], [39] 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 subact 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 enable their inclusion in formal verification. For model checking, an automated translator [17] uses EOFM's formal semantics to convert a given task's XML into the model checker language. This treats each activity and action as a state machine that transitions between ready, executing, and done states. Transitions are based on Boolean conditions created using act strategic knowledge (the pre-, repeat, and completion conditions); the execution state of parent, sibling, and child acts; and the relationships between these as required by the act's position in the task and its parent's decomposition operators. This task state machine model is composed with formal models of the system automation for end-to-end behavioral verification [16].

When erroneous behavior generation is used with EOFM, a version of the formal model is created that enables task formal semantics to be violated in accordance with different theories of human error [20], [21], [40]. This allows model checking to discover how human errors could cause failures.

TABLE I
Equation (1) Parameters for Different Cognitive Functions [44]

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

B. Accounting for Probabilities of Human Errors

Researchers have extended formal human reliability analyses [36], [41], [42], [43] to show that task models can be used with human error rate predictions. The method we focus on here [36] uses a variant [44] of the cognitive reliability error analysis method (CREAM) [45] to dynamically make predictions about human error rates and their impact on system performance. In this, each task (or task part) has an analyst-specified *CPCSum*. This value, 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 is rated as improving (1), reducing (-1), or not affecting (0) human performance. The *CPCSum* is the sum of these ratings. Then, based on a regression model fitted to a large human performance database, human error rates are predicted by Bedford et al. [44]

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

The a, b, and c parameters 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 (functions related to noticing and perceiving events), interpretation (understanding the meaning of the observations as it relates to the current situation), planning (formulating a plan or task to address the current situation), and execution (executing the task). These are used in different parts of task execution and associated with different human errors [36], [44], [45].

When used with EOFM [36], our automatic translator converts an EOFM into a formal model in the input language of PRISM, a state-of-the-art probabilistic and statistical model checker [35]. The formal model was based on the architecture in Fig. 1 (an extension of the one introduced in [16]), where the human task model interacts with the formal representation of the other system elements, which are manually created by

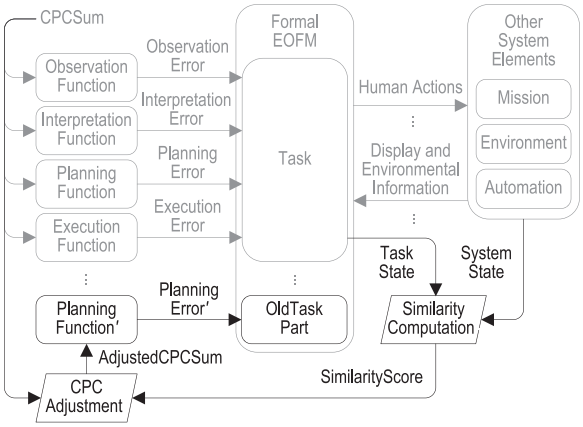


Fig. 1. Architecture used for supporting the prediction of probabilities of human error with EOFM in formal methods analyses. Rounded rectangles describe different model state machines. Parallelograms are formulas that compute values from model variables during execution. Elements in gray show the architecture for the original method [36]. The parts in black show how this architecture was extended to account for negative transfer [37]. Ellipses (⋮) are used to show that there can be additional cognitive functions in a given model, which could be included to influence the execution of different parts of a task. These also signify the possibility of multiple old task parts and associated primed planning functions.

the analyst. Importantly, the formal task drew inputs from concurrent, synchronously composed cognitive function models. These, using a *CPCSum* and (1), indicate if an error in each function can occur (a Boolean variable that becomes true based on the computed probability) 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, use them to determine whether to execute that part of the task normatively or erroneously. The erroneous behavior options were generated based on the task-based taxonomy of human error [30], in accordance with the generation strategy from [40], a generic error generation approach that can encompass deviations from task behavior, including genotypes [1] and phenotypes [2] of human error. Observation function errors would cause inputs to the task model from the other system elements to be perceived incorrectly. Interpretation function errors cause incorrect interpretation of strategic knowledge that dictates when activities execute. Planning function errors result in incorrect execution of task plans (as if the modeled person did not know or properly formulate the task). Finally, execution errors result in the human model omitting, intruding, repeating, or incorrectly performing actions (see details in [36]).

Models created in this way can then be checked to produce probabilistic predictions about error rates and different system outcomes. This approach was able to accurately predict the probability of post-completion errors for different versions of an automated teller machine interface [36] (albeit on the high side of acceptability). However, the method could not account for negative transfer that can occur during a design change.

C. Accounting for Negative Transfer

Negative transfer occurs in a design change when the modifications allow for the old (familiar) behavior to execute under conditions when that execution (previously) would have been appropriate [32], [33]. This can lead to a human performing outdated, incorrect behavior. Specifically, negative transfer arises when there is a combination of surface similarity and structural discrepancy [32]. Surface similarity occurs when the interface or environment conditions are similar enough to trigger the old task. Structural discrepancy indicates that the tasks are executed differently. This results in the human mistakenly executing the old task within the new design. This definition is consistent with the literature that has explored the mechanism and effects of negative transfer [46], [47], [48], which has concluded that negative transfer is predominantly caused by "temporary cognitive or decision confusion, not the result of a motor control problem" [46]. Thus, while negative transfer can occur at any level of cognition [49], it is higher order cognition (such as that associated with the planning cognitive function) that is predominantly responsible for its impact.

We adapted the EOFM language and its translator to incorporate negative transfer into its formal probabilistic analyses [37], thus enabling it to be considered along with its ability to predict human error probabilities through other mechanisms [36]. This was accomplished by: 1) including the replaced parts of the task within the formal task model representation (to capture structural inconsistency); 2) modifying the formal model to dynamically evaluate the surface similarity between the current situation and the original execution conditions of the replaced task; 3) proportionally reducing the CPCSum for the original task based on the maximum potential impact of surface similarity and using this to compute the probability of a specialized planning error causing the replaced task to execute. It is important to note that we treat negative transfer as a unique type of planning error. This is because negative transfer can relate to knowledge about how to perform the task at any point in a task's execution. Thus, the mechanism for the error most naturally and generically relates to the planning function. This is also consistent with the cognitive theory associated with negative transfer discussed above [32], [46], [47], [48], [49].

To enable these changes, the EOFM language and translator were expanded to allow the explicit representation of old tasks in the formal EOFM. For the second and third points, we modified the EOFM-to-PRISM translation process [36] to compute the similarity between the current situation and the conditions under which the original task was performed, dynamically influencing the likelihood that the original task will be executed. The alterations to the formal model are illustrated by the black elements in Fig. 1.

In the revised architecture, when a task component is replaced, the original is retained within the formal EOFM representation as an `Old Task Part`, which is associated with three new elements.

1) `SimilarityComputation` is a formula that calculates a `SimilarityScore` by extracting features from the original condition under which the task part would have been executed and comparing it to the current situation.
2) `CPCAdjustment` is a formula that adjusts the original `CPCSum` downwards, based on the `SimilarityScore`

$$\text{AdjustedCPCSum} = \text{CPCSum} - \Delta \cdot \text{SimilarityScore} \quad (2)$$

where Δ is the maximum possible adjustment to the CPC sum. Given that changes in tasks primarily impact the operator's procedural knowledge, only the planning CPC is adjusted, restricting Δ to a range between 0 and

2 (the actual Δ range would be contingent on how the procedures' CPC was originally assessed).

3) `PlanningFunction'` is a new concurrent cognitive function module that uses the `AdjustedCPCSum` to determine the probability of a planning error occurring as a result of negative transfer (indicated by `PlanningError'`).

If `PlanningError'` is true, the `OldTaskPart` can execute in place of the corresponding portion of the new task. To facilitate this, the formal model marks the new task as completed when `OldTaskPart` has been executed.

The similarity assessment leverages EOFM's ability to provide a formal logical description of when the old task part would execute. This allows for the calculation of surface similarity in accordance with the theory underlying rule-based human cognitive models [50], [51]. This model posits that humans recognize situations where a behavior is appropriate based on the features present, where the set of required features is represented by a Boolean expression. The similarity between the feature set F_A in the current situation and the feature set F_B from the original context is computed as the ratio of shared features to the total number of features in F_B [51]

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

To implement this in EOFM [37], we developed a reasoning engine that transforms inequalities in the Boolean expression into equivalent equalities. Then, the expression is converted into conjunctive normal form, where the features are represented as distinct "or" clauses. The `SimilarityComputations` element (see Fig. 1) uses this structure to calculate the `SimilarityScore`

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

where $\theta(Feature_x)$ equals 1 if feature expression $Feature_x$ is true, and 0 otherwise.

This approach was validated with a case study from the literature for which there was human data: software that simulated packing fruit into boxes with different keyboard keys [52]. The method did predict realistic error rates [37]. However, as with the previous example, these were higher than the actual observed error. Also, the artificiality of the application and the limited size of the data set make it unclear how generalizable the results are.

III. Objectives

To validate our method's predictions, we conducted a human subjects study. In this, we compared the observed error rate of participants for different interface designs produced with our formal method under conditions where negative transfer could occur and when it should not. The evaluated applications were variants of the prescription order entry system used in the OpenEMR electronic medical record system: one based on the actual design and one meant to improve human reliability. This interface is used by medical professionals to electronically send prescriptions to pharmacies. This is an appropriate and important application because errors in prescriptions can cause extra work for pharmacists and medical workers and, if dangerous errors are not caught, critical patient health and mortality outcomes. Furthermore, this is a real system. OpenEMR has more than 20 000 worldwide installations and services approximately 120 million patients [53].

Additionally, we hypothesized that the slight overestimation of error rates observed in previous results was caused by the generic inclusion of planning errors: errors related to formulating a wrong task plan to address an unexpected situation. While such errors are very real [45], they are likely unusual in applications with well-defined and well-practiced task models (which include the applications that were previously evaluated). This is because such systems will have human users who understand and are well-practiced with the system's task procedures. Thus, we created predictions that both included and excluded planning error so that they could be compared with our experimental results.

IV. Methods

The following describes our validation effort. This includes our human subjects experiment, our formal modeling and verification effort, and the data analyses used for assessing the error rates observed in the experiment and comparing them with the predictions made through verification.

A. Human Subjects Experiment

To collect data for our validation, we ran a human subjects study. Participants interacted with two different versions of the OpenEMR prescription order entry interface. This study was conducted under UVA IRB-SBS Protocol 5064.

1) Participants: We recruited 29 nursing students from the University of Virginia as experiment participants (1 male and 28 females); 75% were 18–24 years of age, 7% were 25–29, 14% were 30–34, and 3% were 35–39. All were compensated for their participation with a $50 gift card.

2) Materials and Apparatus: The experiment was conducted in the Human Systems Laboratory at the University of Virginia in a controlled, quiet, and uniformly illuminated setting. It was administered on a laptop computer resting on a computer desk in front of which a participant would sit. The laptop was connected to a mouse and a set of headphones that the participant could use to interact with the computer. The laptop was running software (implemented as a web application built using PHP, JavaScript, and HTML) that was developed specifically for this study. The software was designed to administer the full experiment: introducing participants to the different tasks with training videos, providing them with text instructions; and collecting (and saving) participant responses. The software also included the two prescription order entry interface designs. The first [illustrated in Fig. 2(a)] used the original design from OpenEMR. In the alternative [see Fig. 2(b)], we modified this design to presumably improve the human reliability of the interface. Both had participants enter information concerning whether a prescription was active; its start date; the provider; the name of the drug; the quantity (number of units) of the drug; the medicine unit of the drug (i.e., the amount of the drug in each unit such as a tablet); coded directions for taking the drug (e.g., PO for "per os"/"by mouth" or QD for "quaque die"/"once a day"); the number of refills and associated tablet dosages; prescriber notes; an indicator about whether to add the medication to a particular list; whether substitutions are allowed; whether the prescription was an electronic-prescription (e-prescription); whether it has been checked that the prescription is covered by the patient's insurance (checked drug formulary); and whether the prescription is a controlled substance.

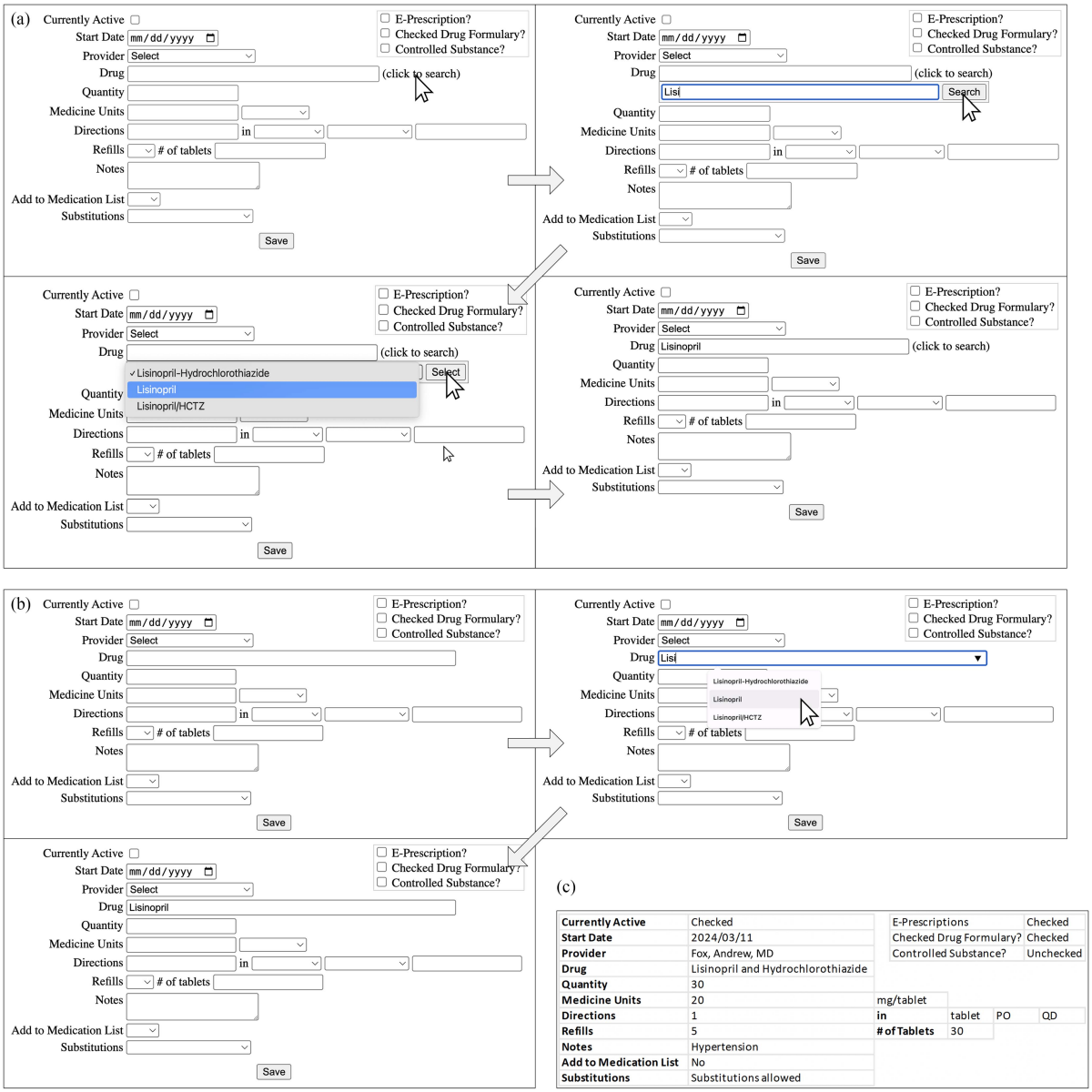


Fig. 2. Interfaces seen by participants in the experiment. (a) and (b) are the two versions of the prescription order entry system from the OpenEMR application. (a) is the original OpenEMR design. (b) is the alternative design. In both (a) and (b), the multiple screenshots and arrows show the process for finding and entering the drug type. (c) is an example of prescription information presented to participants on paper during the experiment.

The differentiating factors between the interfaces were how the drug name was specified. In the original, entering the drug was a multistep process [illustrated in Fig. 2(a)]: 1) the user/participant clicks on the "(click to search)" text next to the "Drug" text field; 2) this displays a search box below the "Drug" field in which the user enters a search term and clicks on the "Search" button; 3) this replaces the search box with a dropdown list containing the search results, from which the user selects the best option and clicks on the (now displayed) "Select" button; which 4) loads the selected drug into the "Drug" text box. Note that this interface allows the user to enter the drug name directly into the "Drug" search box (but without the benefit of input being confirmed with a search). Notably, the original interface violates standard user interaction conventions, which inherently increases the likelihood of interaction errors, even in the absence of negative transfer.

In the alternative interface, we sought to streamline and improve the process for searching for a drug [see Fig. 2(b)]. In this: 1) the user selects and starts typing the name of the drug in the "Drug" text field; 2) as the user types, a search results list appears showing the best matches for the user's text, which the user can select; 3) the selected option is loaded into the "Drug" text field. This was expected to be more reliable (less prone to human error) because it is less complex, more standard-compliant, and avoids creating a post-completion error condition [54]. Specifically, in the original interface, once a user has searched for and found a drug, they must click on the "Select" button to load it into the "Drug" text field. Given that finding or entering the right drug is the user's primary goal, they should be prone to omitting the supplementary, yet critical, goal of clicking the "Select" button.

For both interfaces, the prescriptions entered were given to participants and were printed on individual sheets of paper [see

an example prescription in Fig. 2(c)]. These prescriptions were from a set of real, representative, deidentified prescriptions used for training pharmacy students in a model pharmacy.

3) Independent Variables: The experiment had two independent variables. The interface [original or alternative; see Fig. 2(a) and (b)] was a within-subjects factor. The order participants saw interfaces (original first or alternative first) was a between-subjects factor, where participants entered all of a set of prescriptions first with one interface and then all with the other. This distinction was important because the interfaces could allow negative transfer to occur when going from the alternative (first) to the original (second). This is because participants could enter a prescription drug directly into the "Drug" text field without engaging the search (a situation likely to increase the chance of incorrect entry of the drug's name). The reverse is not true when the alternative interface is second. This is because the original's drug search cannot be executed on the alternative. This combination of variables ultimately created three conditions of interest: 1) the original interface with no negative transfer (the original seen first); 2) the original interface with negative transfer (the original seen second); and 3) the alternative interface (across both orders).

4) Dependent Measures: For each prescription with each interface, participants entered the prescription's information into the interface. Any field whose entered value did not match what was in the prescription (incorrect values, misspellings, or omissions) resulted in the prescription being counted as having an error. This enabled us to compute error rates for each independent variable condition of interest, where the rate was the number of prescriptions with errors for that condition divided by the number of prescriptions for that condition.

Additionally, for each experiment, participants completed a CREAM CPC survey. This used the standard questionnaire established by Hollnagel [45], which asked participants nine questions to assess whether they felt that each of the nine CPCs improved, reduced, or did not affect performance. By counting the responses to this survey, we were able to estimate average CPC sums for both interfaces.

5) Procedure: In the experiment, a participant was admitted to the lab and sat in front of a laptop. The participant had the experiment explained to them. They were also presented with an informed consent document, which they read and signed. Following this, the participant put on the headphones and had the experiment administered to them via our custom software.

In this, participants completed a demographic survey and then watched a training video. The video described the components of prescriptions and how to enter them into the system using the interface that participants would see first. Following the video, participants entered a set of prescriptions (from provided paper printouts) into the first interface. After entering 50 prescriptions, participants completed the CREAM CPC survey for that interface. This completed the first half of the experiment. Participants were given the option to take a short break. When they were ready to continue, the software would present participants with a training video that explained how to use their second interface. The participants would then enter 50 prescriptions into the second interface. This was followed by the CREAM CPC survey for the second interface.

6) Experimental Design: The same 50 prescriptions were used across interfaces and participants. Thus, each participant had to enter each prescription twice: once with the first interface and once with the second. The order of prescriptions was randomized for each interface and participant. Furthermore, the order participants saw interfaces was counterbalanced between them: 15 saw the original first and 14 saw the alternative first.[1]

B. Formal Modeling and Error Rate Prediction

Based on the OpenEMR documentation and our own experience using its online demo, we used EOFM to model the human task behavior for filling prescriptions with our two interfaces. Visualizations of the EOFM are shown in Figs. 3 and 4. In particular, the general form of the tasks for entering a prescription is shown in Fig. 3(a) along with general patterns for entering text/numerical values [see Fig. 3(b)] and selecting check box options [see Fig. 3(c)]. The variants of the `aEnterDrug` activity [a subactivity of the task in Fig. 3(a)] for entering the name of the drug are shown in Fig. 4. Fig. 4(a) was the activity for entering the drug name into the original interface, while Fig. 4(b) was the activity for doing it with the alternative.

The main task [see Fig. 3(a)] has root activity `aEnterPrescription`. This is executed by having the human enter a prescription's data (`aEnterData`) and then save it (`aSaveData`) as dictated by the `ord` decomposition operator. Data entry under `aEnterData` had activities for each interface field, and could be executed in any order (as per the `and_seq` decomposition). Entering a numerical or textual value used the pattern in Fig. 3(b). This had humans read the associated value from the paper prescription and then set the value (and repeatedly try to do so until the correct value was entered) by selecting the correct interface field and entering the read value. The activity for setting a checkbox value [see Fig. 3(c)] operated similarly, with the participant reading the prescription element from the paper and setting the check value by clicking the appropriate field.

The activity for setting the drug name with the original interface [see Fig. 4(a)] had the human start by reading the drug from the provided paper (`aReadDrug`). They would then set the drug (repeatedly if necessary via `aSetDrug`) by: showing the search field by clicking on "click to search" (under `aSelectClickToSearch`), selecting the revealed search field (with `aSelectSearchBox`), entering the name of the drug they read from the paper (under `aEnterDrugName`), clicking the "Search" button (via `aSelectDrugSearchField`, selecting the revealed search box dropdown list (with `aSelectDrugName`), picking the drug they read from the paper from the list of search results (under `apickDrugname`), and then clicking the "Select" button to load the selected drug into the "Drug" text field (via `aSolidifyDrugnameSelection`). The comparable drug name selection activity for the alternative interface [see Fig. 4(b)] was comparatively simpler. The human reads the drug name from the provided paper (`aReadDrug`) and then sets the drug (again repeatedly if necessary) by: selecting the drug name field (`aSelectDropdownField`), entering the name of the drug read from the paper (`aEnterDrugName`), and then (optionally via the `optor_seq` decomposition) picking the drug name from the search results (*aPickDrugName*).

These tasks were translated into PRISM's input language to create six different models. Two were for the original interface [using `aEnterDrug` from Fig. 4(a)], one with and one without

[1] Our original design called for 15 participants in each condition. However, participant cancellation and semester schedule constraints resulted in us failing to recruit our final participant.

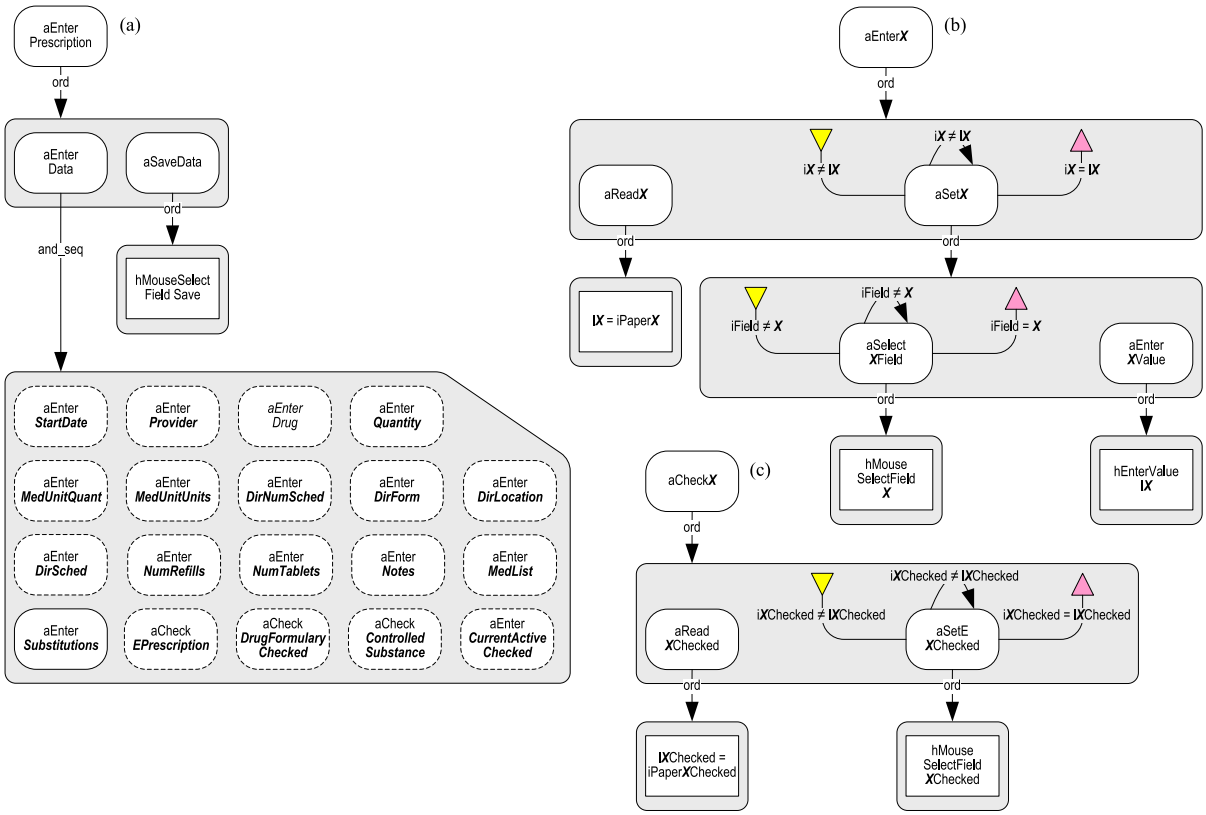


Fig. 3. Visualization of the EOFM patterns for entering prescriptions into the OpenEMR prescription order entry interfaces (see Fig. 2). In these patterns (and those in Fig. 4), activities are rounded rectangles and actions are pointy ones. Decompositions are downward arrows annotated with a decomposition operator. These point to a gray shape that encompasses decomposed acts. Strategic knowledge conditions are connected to the activity they constrain. These are labeled with the condition's Boolean logic. A *Precondition* is a down-pointing, yellow triangle; a *CompletionCondition* is an up-pointing, magenta triangle; and a *RepeatCondition* is a recursive arrow. In these examples, there are two types of action. Local actions have the person remember a value by setting a local variable (one starting with `l`) to a value (e.g., `l`**X**`= iPaper`**X**). Valued output actions communicate a value of a variable (e.g., `hMouseSelectField`**X**). See [55] for more details about the notation. (a) Base EOFM for the original and modified open EMR interfaces. Dotted activities are defined in other sub-figures or Fig. 4. Italicized and bolded names in activities indicate the values of variable **X**, which are used to define instances of the patterns in the other subfigures. (b) Activity pattern (based on **X**) for setting a value field on the interfaces. (c) Pattern (based on **X**) for setting an interface checkbox.

standard planning errors, where planning errors were excluded by manually setting `PlanningError` (see Fig. 1) to false. Two versions were for the original interface in the negative transfer conditions using `aEnterDrug` from Fig. 4(a), both with and without standard planning errors, where `aSetDrug` from Fig. 4(b) was the old task part. Note, negative transfer planning errors, using `planningFunction'` from Fig. 1, were included in the negative transfer condition. Furthermore, the similarity condition (4) that could contribute to errors associated with executing Fig. 4(b)'s `aSetDrug` as the old text part was formulated as $(\theta(aEnterDrug = Executing) + \theta(aReadDrug = Done) + \theta(iDrugName \neq lDrugName))/3$. The final two model versions were for the alternative interface [using `aEnterDrug` from Fig. 4(b)], one with and one without standard planning errors. In these, the architecture's (see Fig. 1) other system elements were composed of automation and mission models.[2] The automation model tracked the interface field that was selected, updated field values based on this, and (in the case of the original interface) controlled the display and selection of fields for the drug name search. The mission model would determine what an accurate prescription looked like, where the probability of a given prescription form was set based on the set of prescriptions participants entered during the experiment. Note, to keep the model tractable, all prescription values were modeled abstractly as being `Nothing` (having no value set), `SomethingCorrect` (having a value that matches what was in the prescription), and `SomethingIncorrect` (having a value not matching what was in the prescription).

Finally, the formal models made use of PRISM's "label" feature to define when a prescription error occurred under the label "RxError." This label logically computed a Boolean value that would be true if the `aEnterPrescription` [see Fig. 3(a)] completed execution with the entered prescription not matching what was prescribed in the mission. It was false otherwise.

By default, analyses for all models started with the `CPCSum` set to the maximum value of 9. This was because, in our expert opinion, we expected the task to be fairly familiar and straightforward for most participants, who were performing the task in an ideal environment. However, we also potentially anticipated updating these to match the interface condition average `CPCSum` measured from participant surveys.

Each model variation was verified using statistical model checking (to avoid the scalability issues of probabilistic checking; [40]) with confidence set to 0.01, 20 000 samples, and a maximum path length of 10 000. These used a probabilistic

[2] All models and software can be found at https://github.com/mlb4b/PEOFM-to-Prism-Translator.

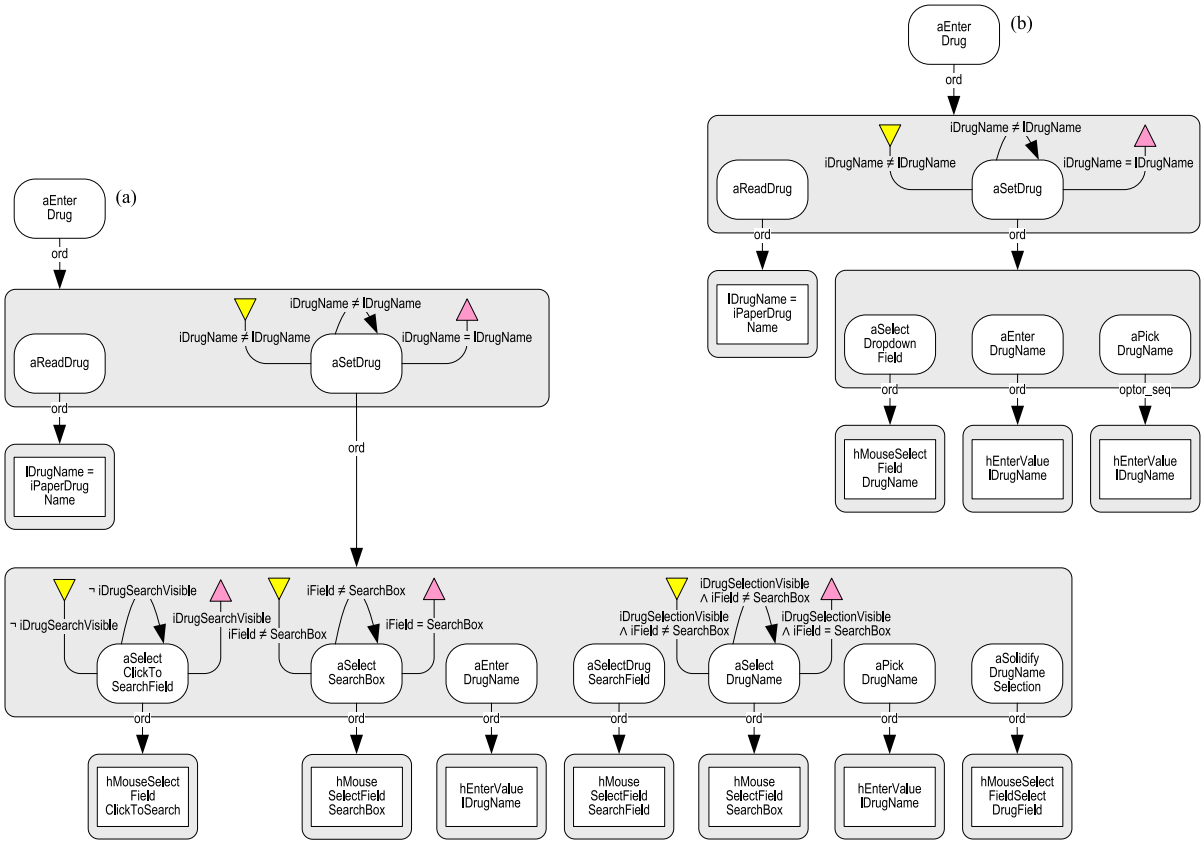


Fig. 4. EOFM activities for entering the drug (aEnterDrug from Fig. 3) for the (a) original and (b) modified open EMR interfaces. See Fig. 3 for a description of the notation.

computation tree logic property of

$$P = ?\,[F(\text{“RxError”})]. \tag{5}$$

This instructed the model checker to compute the probability ($p = ?$) that eventually (F) “RxError” becomes true.

C. Data Analysis

We used two-tailed statistical tests to evaluate whether there were significant differences observed between options using $\alpha = 0.05$, with p-values adjusted using a Benjamini-Hochberg (BH) false discovery rate (FDR) correction to account for multiple comparisons. This included a paired t-test to assess whether there were differences in the error rates observed between the original and alternative interfaces. It also included independent t-tests to evaluate whether there were differences between the negative transfer conditions for the experimentally observed error rates for the original interface. Finally, it did not make sense to use a t-test to compare the observed human error rates to those produced by the model verification process. This is primarily due to the former being averaged across observations collected from multiple participants and the latter being computed based on probabilistic computations from 20 000 model traces. However, both processes produce accurate estimates of sampling variability. Thus, we used Z-tests for the difference between two proportions to assess whether there were significant differences between the predicted error rates (both with and without standard planning errors) and their corresponding interface’s experimental error rate.

V. Results

Based on the ratings participants provided on the CPC survey about entering prescription data with the different interfaces, the original interface received a mean CPCSum of 2.510 ($sd = 1.844$). The alternative interface received a mean CPCSum of 1.385 ($sd = 1.500$). These values were considerably lower than our expert-provided CPCSum of 9, which we initially used for model predictions. Because this expert CPCSum produced extremely accurate results (discussed next), we did not use the averages provided by participants (which would have produced significantly higher error rate predictions) or explore additional CPCSum levels.

The error rates observed in the experiment were 0.054 for the original interface (without negative transfer / when seen first), 0.056 for the original interface under the negative transfer condition (when seen second); and 0.033 for the alternative interface (an average of a 0.033 error rate observed when the alternative interface was seen both first and second). These were compared with the results produced by each of the evaluated models with a CPCSum of 9, as shown in Fig. 5.

The statistical analyses of these results revealed significant differences between the experimentally observed error rates for the original and alternative interfaces ($t(28) = 26.10, p = 0.033, d = 4.85$) and the experimental error rate and the predicted one produced when planning errors were allowed ($z = 3.06, p = 0.018, h = 0.11$). No other comparisons showed significant differences. These results are summarized as part of Fig. 5. There are three important takeaways from these results.

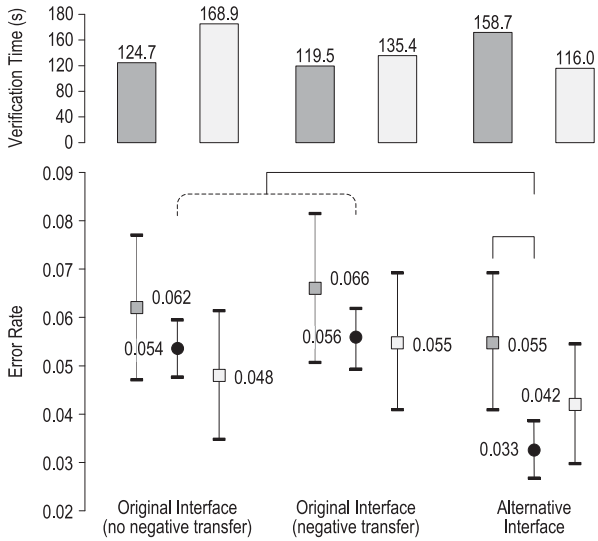


Fig. 5. Error rates observed from the human subject experiment (•) compared with model verification results both with (■) and without (□) standard planning errors allowed. All results are shown with 95% confidence intervals observed across participants or produced by verification (converted from the 99% intervals produced by the model checker). The brackets above the plots show significant differences between options ($p < 0.05$ adjusted using BH FDR). A dotted, curved bracket indicates a pooling between options. Verification times are reported above their respective model error rate predictions.

First, the alternative interface was clearly more reliable (produced a significantly lower error rate) than the original in the human subjects experiment. Second, while the negative transfer condition for the original interface produced a higher error rate, this was not significantly different from the condition that did not have negative transfer. Third, the error predictions that excluded standard planning errors were closer to the comparable error rates observed in the experiment. This relationship was most clear in the alternative interface condition. Here, predictions that included standard planning errors were significantly higher than the experimental results, while those performed without the planning errors were not.

VI. Discussion

The results of this research appear to validate the ability of our method to accurately predict human error rates. Using the analyst identified `CPCSum` of 9, our approach was able to (with one exception) produce error estimates that were within 1% point of the observed values and whose 95% confidence interval encompassed the observed value (see Fig. 5). The exception was the alternative interface when planning errors were allowed, which was within 2.2% points of the experimental result. Additionally, all the predicted values reproduced the ordinality of the error rates observed for the different interface conditions (within a given planning/no-planning option): the highest error rate was seen with the original interface under the negative transfer condition, the second highest with the original without negative transfer, and the lowest with the alternative interface.

Furthermore, the results appear to confirm our hypothesis that overestimation of error rates was caused by the generic inclusion of planning errors (planning errors not associated with negative transfer) in predictions (see Fig. 5). Specifically, all the error rate predictions were overestimates (larger by between 0.8 and 2.2 percentage points) when planning errors were included. When they were excluded, error rates were both closer (within 0.1% and 0.9% points), ranged around the observed value, and were not significantly different from those values. Thus, our recommendation would be to use the method without including generic planning errors unless an analyst has a compelling reason to do so.

The experiment did produce a higher error rate with the original interface when negative transfer was expected. However, the difference observed was only 0.2% points (∼0.2 errors per participant): not a statistically significant change. This was a surprising result due to the strong negative transfer condition created for the experiment and the prominence of the claim that this is a major factor in human reliability in the human factors and usability literature, particularly in the work of Norman [34]. It is possible this was caused by the simplicity of the task and/or the fact that the associated error could only manifest in limited ways.

Based on our insignificant finding, we returned to the literature to see how our observed rates compared to others'. Besnard and Cacitti [52] trained participants in a simple computer interface for sorting virtual fruit based on specific keyboard keys. When they swapped around the keys, they observed an average, small increase of 1.6 errors (they did not report rates or statistical tests). Woltz et al. [56] reported three studies where error rates were tracked in negative transfer conditions. In these, participants repeatedly reduce numbers using a given mental algorithm. Participants in the negative transfer conditions were heavily trained with a particular set of numbers and then asked to perform with a different set. They produced significantly more errors than those who did not receive the extensive training. Blais et al. [46] had participants perform a coordination task where they were asked to move a 2-D joystick and a foot control to the position of stimuli on a visual display. All participants reviewed consistent initial training using standard controls, had their controls reversed for a variable number of trials, and then were evaluated when controls were unreversed. The researchers found that an increase in the number of reversed control trials participants experienced resulted in more negative transfer errors when the controls were ultimately unreversed. The observed negative transfer error rates ranged from 0.75% to 2.25% .

These results, coupled with our own, suggest that negative transfer may only be relevant to particular types of tasks. Besnard and Cacitti's [52] study related to keying errors in response to a well-practiced stimulus-and-response sequence; Woltz et al.'s [56] concerned sequential problem solving; Blais et al.'s [46] was a manual control task; and ours was data entry. Thus, more research is required to understand in what types of tasks negative transfer is a critical factor in human reliability. It is important to note that our previous results [37] did capture the negative transfer effect observed by [52]. This, coupled with our results here, suggests (though clearly not definitively) that our method is capable of predicting the minor effect of negative transfer on keyboard-related human–computer interaction tasks. Future research should investigate whether this extends to other human–computer interaction tasks as well as the tasks explored by Woltz et al. [46], [56], and others in the negative transfer literature.

Although not used in the work presented here, the PRISM model checker can compute rewards/penalties based on the state of the model. This can allow the model to count the number of errors across multiple task executions [37]. This feature could theoretically be used (with multiple reward or penalty

categories) to count different categories of errors or specific instances of error to provide more diagnostic information about predicted error rates. The implementation and validation of such a feature should be explored in future research.

The results of our study do indicate that the alternative interface design we implemented for the drug search in OpenEMR's prescription order entry interface was superior: it produced an error rate 2% lower than the current one. We encourage the open EMR developers to implement such an interface in their system to potentially improve patient safety.

It is standard practice in CREAM that CPCs are assessed by subject-matter experts, ones that understand both the domain and CREAM [45]. The results we obtained from our CREAM CPC survey suggest that normal system users are not reliable at estimating these values. Thus, we used our own expert ratings when making model predictions. In particular, the participant-provided ratings not only suggest poorer support from the alternative interface (which contradicted our instincts, model predictions, and empirical results), but both interfaces received estimates that would predict error rates orders of magnitude larger than what we achieved with our estimate. Future work should investigate if and how CPC ratings can be effectively collected from standard users. In the meantime, the standard practice of using experts is recommended.

The results in this article add to the evidence [36], [37] that our method, with or without negative transfer, accurately predicts human error rates and their impact on safety in complex, dynamic systems. Future work should explore its applicability to other safety-critical domains with more complicated tasks. In particular, research should identify domains where planning errors are relevant to see if our method produces accurate predictions for them. Furthermore, future research should explore applications where interface and working conditions are associated with lower CPC ratings to see if the method (with or without negative transfer) generalizes to such conditions.

REFERENCES

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

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

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

[4] R. Kebabjian, "Accident statistics," 2023. [Online]. Available: planecrashinfo.com

[5] AOPA, "The Richard G. mcspadden report: 34th AOPA air safety institute accident report," *AOPA*, Air Safety Institute, Frederick, MD, USA, Tech. Rep., 2025. [Online]. Available: https://www.aopa.org/training-and-safety/air-safety-institute/accident-analysis/richard-g-mcspadden-report

[6] 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 Res. Lab, Adelphi, MD, USA*, Tech. Rep. 2004-11, 2004.

[7] M. A. Makary and M. Daniel, "Medical error—the third leading cause of death in the us," *BMJ*, vol. 353, 2016, Art. no. i2139.

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

[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 Trans. Syst., Man Cybernetics: Syst.*, vol. 43, no. 3, pp. 488–503, May 2013.

[10] M. L. Bolton, "Novel developments in formal methods for human factors engineering," in *Proc. Hum. Factors Ergonom. Soc. Annu. 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, Germany: Springer, 2017.

[12] T. E. Wang and A. Pinto, "Survey of human models for verification of human-machine systems," 2023, *arXiv:2307.15082*.

[13] M. L. Bolton and W. D. Gray, *Cognitive Modeling for Cognitive Engineering* (Ser. Cambridge Handbooks in Psychology). Cambridge, U.K.: Cambridge Univ. Press, 2023, pp. 1088–1112.

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

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

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

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

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

[19] R. E. Fields, "Analysis of erroneous actions in the design of critical systems," Ph.D. dissertation, Dept. of Computer Science, Univ. York, New York, USA, 2001.

[20] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Generating phenotypical erroneous human behavior to evaluate human-automation interaction using model checking," *Int. J. Hum.- Comput. Stud.*, vol. 70, no. 11, pp. 888–906, 2012.

[21] 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 Trans. Syst., Man Cybern.: Syst.*, vol. 43, no. 6, pp. 1314–1327, Nov. 2013.

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

[23] D. Pan and M. L. Bolton, "Properties for formally assessing the performance level of human-human collaborative procedures with miscommunications and erroneous human behavior," *Int. J. Ind. Ergonom.*, vol. 63, pp. 75–88, 2018.

[24] M. L. Bolton, "Model checking human–human communication protocols using task models and miscommunication generation," *J. Aerosp. Inf. Syst.*, vol. 12, no. 7, pp. 476–489, 2015.

[25] A. Barbosa, A. C. Paiva, and J. C. Campos, "Test case generation from mutated task models," in *Proc. 3rd ACM SIGCHI Symp. Eng. Interactive Comput. Syst.*, 2011, pp. 175–184.

[26] M. L. Bolton, S. S. Taylor, and L. Humphrey, "A formal method for assessing mental workload," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2023, pp. 5255–5260.

[27] P. Wan and M. L. Bolton, "A taxonomy of forcing functions for addressing human errors in human-machine interaction," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2021, pp. 3134–3139.

[28] C. Fayollas, C. Martinie, and P. Palanque, "Taking into account human error when assessing the impact of dependability on usability," in *Proc. IEEE 35th Int. Symp. Softw. Rel. Eng. Workshops*, 2024, pp. 271–278.

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

[30] M. L. Bolton, "A task-based taxonomy of erroneous human behavior," *Int. J. Hum.- Comput. Stud.*, vol. 108, pp. 105–121, 2017.

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

[32] P. Johnson, "Supporting system design by analyzing current task knowledge," in *Task Analysis for Human-Computer Interaction*, D. Diaper, Ed. Chichester, U.K.: Ellis-Horwood, 1989, pp. 160–185.

[33] D. N. Perkins and G. Salomon, "Transfer of learning," *Int. Encyclopedia Educ.*, vol. 2, pp. 6452–6457, 1992.

[34] D. A. Norman, *The Psychology of Everyday Things*. New York, NY, USA: Basic Books, 1988.

[35] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. Int. Conf. Comput. Aided Verification*, 2011, pp. 585–591.

[36] M. L. Bolton, X. Zheng, and E. Kang, "A formal method for including the probability of erroneous human task behavior in system analyses," *Rel. Eng. Syst. Saf.*, vol. 213, 2021, Art. no. 107764.
[37] M. L. Bolton, S. Riabova, Y. Son, and E. Kang, "Negative transfer in task-based human reliability analysis: A formal methods approach," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2023, pp. 5249–5254.
[38] B. Kirwan and L. K. Ainsworth, *A Guide to Task Analysis*. London, U.K.: Taylor and Francis, 1992.
[39] 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, Switzerland: Springer, 2017, pp. 343–377.
[40] 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," *Rel. Eng. Syst. Saf.*, vol. 188, pp. 168–180, 2019.
[41] X. Zheng, M. L. Bolton, C. Daly, and L. Feng, "A formal human reliability analysis of a community pharmacy dispensing procedure," in *Proc. Hum. Factors Ergonom. Soc. Annu. Meeting*, 2017, pp. 728–732.
[42] X. Zheng, M. L. Bolton, C. Daly, and E. Biltekoff, "The development of a next-generation human reliability analysis: Systems analysis for formal pharmaceutical human reliability (SAFPH)," *Rel. Eng. Syst. Saf.*, vol. 202, 2020, Art. no. 106927.
[43] X. Zheng, M. L. Bolton, and C. Daly, "Extended SAFPH (systems analysis for formal pharmaceutical human reliability): Two approaches based on extended CREAM and a comparative analysis," *Saf. Sci.*, vol. 132, 2020, Art. no. 104944.
[44] T. Bedford, C. Bayley, and M. Revie, "Screening, sensitivity, and uncertainty for the CREAM method of human reliability analysis," *Rel. Eng. Syst. Saf.*, vol. 115, pp. 100–110, 2013.
[45] E. Hollnagel, *Cognitive Reliability and Error Analysis Method (CREAM)*. Oxford, U.K.: Elsevier, 1998.
[46] C. Blais, R. Kerr, and K. Hughes, "Negative transfer or cognitive confusion," *Hum. Perform.*, vol. 6, no. 3, pp. 197–206, 1993.
[47] R. A. Schmidt, "A schema theory of discrete motor skill learning," *Psychol. Rev.*, vol. 82, no. 4, pp. 225–260, 1975.
[48] R. A. Schmidt and D. E. Young, "Transfer of movement control in motor skill learning," in *Transfer of Learning*. Amsterdam, The Netherlands: Elsevier, 1987, pp. 47–79.
[49] M. L. Gick and K. J. Holyoak, "The cognitive basis of knowledge transfer," in *Transfer of Learning*. Amsterdam, The Netherlands: Elsevier, 1987, pp. 9–46.
[50] A. Tversky, "Features of similarity," *Psychol. Rev.*, vol. 84, no. 4, pp. 327–352, 1977.
[51] R. Sun, "Robust reasoning: Integrating rule-based and similarity-based reasoning," *Artif. Intell.*, vol. 75, no. 2, pp. 241–295, 1995.
[52] D. Besnard and L. Cacitti, "Interface changes causing accidents. an empirical study of negative transfer," *Int. J. Hum.- Comput. Stud.*, vol. 62, no. 1, pp. 105–125, 2005.
[53] P. Groen, "OpenEMR continues to grow in popularity and use," *Open Health News*, Dec. 18, 2012. [Online]. Available: https://www.openhealthnews.com/hotnews/openemr-continues-growpopularity-and-use
[54] P. Curzon and A. Blandford, "Formally justifying user-centered design rules: A case study on post-completion errors," in *Proc. 4th Int. Conf. Integr. Formal Methods*, 2004, pp. 461–480.
[55] M. L. Bolton and E. J. Bass, "Using task analytic models to visualize model checker counterexamples," in *Proc. IEEE Int. Conf. Systems, Man, Cybern.*, 2010, pp. 2069–2074.
[56] D. J. Woltz, M. K. Gardner, and B. G. Bell, "Negative transfer errors in sequential cognitive skills: Strong-but-wrong sequence application," *J. Exp. Psychol.: Learn., Memory, Cogn.*, vol. 26, no. 3, pp. 601–625, 2000.

Yeonbin Son (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in industrial and management engineering from Kyonggi University, Suwon, South Korea, in 2018 and 2020, respectively. She is currently working toward the Ph.D. degree in systems engineering with the Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA, USA.

Her research interests include machine learning-based recommender systems and the use of human performance modeling.

Matthew L. Bolton (Senior Member, IEEE) received the B.S. degree in computer science, the M.S. degree in systems engineering, and the Ph.D. degree in systems engineering from the University of Virginia (UVA), Charlottesville, VA, USA, in 2004, 2006, and 2010, respectively.

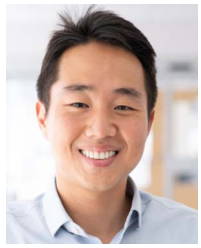
He is currently an Associate Professor with the Department of Systems and Information Engineering, UVA. His research interests include the use of human performance modeling and formal methods in the design and analysis of safety-critical systems.

Emma Crooks received the B.S. degree in industrial engineering from the University at Buffalo, Buffalo, NY, USA, in 2021.

She is a Project Engineer with the US Army Corps of Engineers.

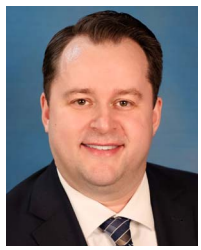
Hannah Palmer received the B.S. degree in systems and information engineering from the University of Virginia, Charlottesville, VA, USA, in May 2025.

She will be joining General Dynamics Mission Systems, Chantilly, VA, USA, as a Systems Engineer. Her academic interests are in human–computer interactions.



Eunsuk Kang received the B.S.E. degree in software engineering in 2007 from the University of Waterloo, Waterloo, CA, USA, the S.M. and Ph.D. degrees in computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2010 and 2016, respectively.

He is an Associate Professor with the Software and Societal Systems Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include rigorous modeling and analysis techniques to design safe, secure, and reliable software.



Christopher Daly received both the PharmD and MBA degrees from the University at Buffalo School of Pharmacy and Pharmaceutical Sciences in Buffalo, NY, USA in 2012.

He is a Clinical Associate Professor with the University at Buffalo School of Pharmacy and Pharmaceutical Sciences, Department of Pharmacy Practice, Division of Outcomes and Practice Advancement. His professional efforts support the growth and sustainability of patient care services within community-based pharmacy settings. His academic and research interests include innovative outpatient pharmacy models, entrepreneurialism, social and administrative pharmacy practice sciences, and clinical-based outcomes.