Probabilistic Models for the Verification of Human-Computer Interaction

Bernhard Beckert and Markus Wagner

Department of Computer Science, University of Koblenz, Germany
beckert@uni-koblenz.de, wagnermar@uni-koblenz.de

Abstract. In this paper, we present a method for the formalization of probabilistic models of human-computer interaction (HCI) including user behavior. These models can then be used for the analysis and verification of HCI systems with the support of model checking tools. This method allows to answer probabilistic questions like "what is the probability that the user will unintentionally send confidential information to unauthorized recipients." And it allows to compute average interaction costs and answer questions like "how much time does a user on average need to send an email?"

1 Introduction

Interaction between computer and user is a two-way communication process, where the user enters commands and the system responds to the input. This is referred to as interactive control [1]. Unfortunately, this control does not always work perfectly and errors in human-computer interaction (HCI) occur. A huge number of errors in HCI can be traced back to user interfaces that are insufficiently secured against these errors, irrespective of whether the error is caused (a) by the user – intentionally or unintentionally – interacting incorrectly with the computer, (b) the computer reacting incorrectly to the user's input, or (c) the user interpreting the computer's output incorrectly. The consideration of user errors and their overall impact on the system forms an important part of an analysis of a system's usability, safety, and security. As a result, the designers of systems often consider human frailty and try to reduce errors and usability problems. A popular approach is to stick to informal lists of design rules [1–3].

In our approach to the analysis of HCI, three model components are combined: (1) a model of the user's behavior, (2) a model of the system's behavior, and (3) a model of the user's assumptions about the system's state (based on the user's inputs and the systems reactions). The advantage of our method is that each model component can be probabilistic. This allows for the specification of errors in the user behavior, in the application, and in the way the user interprets the application's reactions. The goal of our technique is to prove properties of the interaction by means of probabilistic model checking – or more specifically, to formally prove that, considering all the possible ways of HCI in a given scenario, the given probabilistic requirements are fulfilled.

The basis of our work is the (non-probabilistic) method of Beckert and Beuster [4] for the formalization, analysis, and verification of user interfaces. It is based on GOMS [5], which is a well-established user modeling method. The GOMS models are formalized and augmented with formal models of the application and the user's assumptions about the application's state.

2 Cost and Probability of HCI Sequences

Human-computer interactions can be split into sequences of actions. In our approach, each action of the user or (re-)action of the system is represented by a transition in the HCI model. To take the different cost of actions into consideration, the notion of *action cost* is introduced, which can be used to represent quantities like "amount of time", "amount of money," or "amount of resources." Action costs are attached to the corresponding transition in the model. Later, this is used to reason about the effects of minor errors, such as mistyping characters when writing an email, and major errors, such as sending confidential data to unauthorized recipients. Whether an interaction is an error or not depends on what the goal of the interaction is. Given a set of goal states, an action is an error if it increases the total cost required to reach the goal or makes it impossible to reach the goal. In the latter case, the error cannot be undone, in the former case, additional or more costly actions are required to undo the effect of the error, i.e., an *error cost* is introduced.

Based on the total interaction cost and on the assumption that there is a limit on the acceptable cost of the interaction, the sequences of actions that start in the initial state and end in a goal state can be classified into three categories:

1. optimal sequences with minimal total cost,
2. acceptable sequences with total cost that do not exceed the limit, and
3. unacceptable sequences with total costs that exceed the user's budget.

The analyst of an HCI scenario may be interested in computing the probabilities or the expected cost of reaching certain states. This can be, e.g., a single state, a set of goal states, or all the final states of the interaction. Similarly, the analyst may be interested in verifying that certain states, or set of states, are (not) reached with a certain probability. With our method, this all can be done.

Regarding the action's probabilistic values that are needed for the construction of the interaction model, the analyst applying our method may use estimates or experimental data. Also, databases containing probabilities of different kinds of human errors may be used (e.g. [6, 7]). This data comes from many sources, such as nuclear power plants, simulator studies, and laboratory experiments. A method is provided for combining the data in order to produce estimations of the erroneous executions of tasks.

3 Basis: The Non-Probabilistic Model

In this section, we present the non-probabilistic basis of our model. Following [4], we assume that a user interacts with a system based on what his or her

assumptions of the application's configuration are, including assumptions about the internal state and relevant data. In terms of Linear Temporal Logic (LTL), always correctly interprets the system's state if:

$$\mathbf{G}((a_0 \leftrightarrow c_0) \wedge (a_1 \leftrightarrow c_1) \wedge \cdots \wedge (a_n \leftrightarrow c_n)) \ ,$$

where $a_0, \ldots, a_n$ are the critical properties of the application, and $c_0, \ldots, c_n$ are the user's assumptions about whether these properties hold or not.

If this formula does not hold, the user is error-prone. Then, the following scenarios are possible: (1) Parts of the user's assumptions are wrong, and (2) the user's assumptions are incomplete (over abstraction). For example, in the first scenario the user could assume that the software is in another state that allows other actions to be executed. In the second scenario, the user could be unaware of the necessity to take certain action to avoid high costs. But even if the above formula holds, the user can still execute erroneous actions. Even though he or she could have known better (having the right assumptions about the system state), missing knowledge about what the most cost-effective actions are may lead to errors.

In [4], the model of the interaction is the result of the combination of three model components:

1. a formal GOMS model describing the user behavior,
2. a component representing the user's assumptions of the software's state, and
3. a component representing the application itself.

Input Output Labeled Transition Systems (IOLTS) are used to model these components.

Definition 1. *An IOLTS is a tuple* $L = (S, \Sigma, s_0, \rightarrow)$ *where* S *is a set of* states, $s_0 \in S$ *is an* initial state, $\rightarrow \subseteq S \times \Sigma \times S$ *is a* transition relation, *and* Σ *is a set of* labels *with* $\Sigma = \Sigma? \cup \Sigma! \cup \Sigma I$. *We call* $\Sigma?$ *the* input alphabet, $\Sigma!$ *the* output alphabet, *and* ΣI *the* internal alphabet.

For example, whenever the user executes an action, the corresponding state transition is performed. The corresponding edge in the automaton is annotated with a label denoting that action.

The transition systems are combined by *mutual composition*. In mutual compositions of IOLTSs L_a and L_b, the output of L_a serves as input for L_b, and the output of L_b serves as input of L_a, which is illustrated in Figure 1.

Definition 2. *Let* $L_a = (S_a, \Sigma_a, s_{0a}, \rightarrow_a)$ *and* $L_b = (S_b, \Sigma_b, s_{0b}, \rightarrow_b)$ *be two IOLTSs. We assume the input and output alphabets of* L_a *and* L_b *to consist of internal and external subsets, where the internal input is denoted with* $\Sigma?I$, *the external input with* $\Sigma?E$, *the internal output with* $\Sigma!I$, *and the external output with* $\Sigma!E$. *And we demand that these subsets are chosen such that* $\Sigma!I_a = \Sigma?I_b$ *and* $\Sigma!I_b = \Sigma?I_a$. *Then, the* mutual composition $(L_a||_m L_b) = (S, \Sigma, s_0, \rightarrow)$ *of*

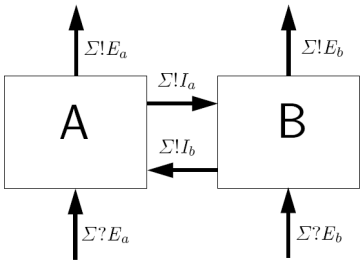

Fig. 1. Mutual composition of two IOLTSs

L_a and L_b is defined by:

$$
\begin{aligned}
S &= S_0 \times S_1 \\
\Sigma? &= \Sigma?E_a \cup \Sigma?E_b \\
\Sigma! &= \Sigma!E_a \cup \Sigma!E_b \\
\Sigma I &= \Sigma I_a \cup \Sigma I_b \cup \Sigma!I_a \cup \Sigma!I_b \\
s_0 &= (s_{0a}, s_{0b}) \\
\rightarrow &= \{(s_a, s_b), \sigma, (s'_a, s_b)) \mid s_a \xrightarrow{\sigma}_a s'_a \ \textit{with}\ \sigma \in \Sigma?E_a \cup \Sigma!E_a \cup \Sigma I_a\} \cup \\
&\quad \{(s_a, s_b), \sigma, (s_a, s'_b)) \mid s_b \xrightarrow{\sigma}_b s'_b \ \textit{with}\ \sigma \in \Sigma?E_b \cup \Sigma!E_b \cup \Sigma I_b\} \cup \\
&\quad \{(s_a, s_b), \sigma, (s'_a, s'_b)) \mid s_a \xrightarrow{\sigma}_a s'_a \ \textit{and}\ s_b \xrightarrow{\sigma}_b s'_b \ \textit{with} \\
&\qquad\qquad \sigma \in \Sigma!I_a \cup \Sigma!I_b = \Sigma?I_b \cup \Sigma?I_a\}
\end{aligned}
$$

The mutual composition of the three aforementioned model components provides the complete model of the interaction, making complete formal modeling possible (see Figure 2).

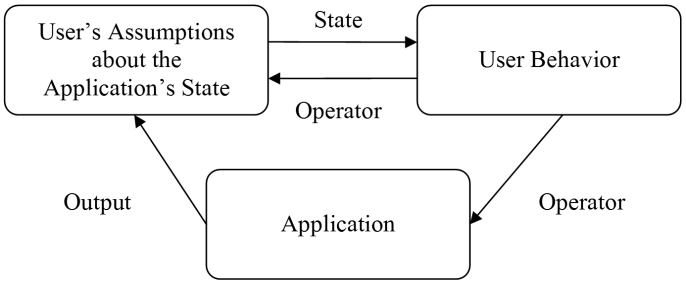

Fig. 2. Basic model of the interaction.

4 The Probabilistic Extension

For our probabilistic extension of the models described in the previous section, we introduce several modifications. In order to incorporate probabilistic values as well as the idea of costs, variations of the IOLTS are introduced. Thus, the state transitions of the resulting models contain information about the actions' costs and the actions' probabilities.

A *Probabilistic Input Output Labeled Transition System* (PIOLTS) is an IOLTS, where the set Σ of labels consists of tuples (l, p) where l is the label denoting the action and p is the probability of that action ($0 \leq p \leq 1$). Similarly, a *Valued Input Output Labeled Transition System* (VIOLTS) is an IOLTS, where Σ consists of tuples (l, v) where v is the cost assigned to that action ($v \geq 0$). Finally, a *Probabilistic Valued Input Output Labeled Transition System* (PVIOLTS) is a combination of both variants, where Σ consists of tuples (l, p, v).

To model the user's behavior, we use a PVIOLTS. Thus, not only the probabilistic behavior is modeled, but also "personal" costs. The application's model is extended to a VIOLTS by numerical values representing the costs for the execution of each single step. Probabilities can be added to model non-deterministic application behavior. The user's assumptions are modeled using a PIOLTS, and costs can be added if needed. As sources for the statistical data, we suggest to use databases as mentioned in Section 2. However, this is not mandatory and the analyst can come up with his/her own values. Now, the mutual composition of these three components provides the extended model of the HCI, incorporating probabilistic user behavior as well as the interaction cost.

The probabilistic models resulting from our method can be used to prove quantitative aspects of the interaction. In general, probabilistic model checking is an automatic formal verification technique for the analysis of systems that exhibit stochastic behavior [8]. In our setting, the models can be represented as Discrete-Time Markov chains. Using Markov chains instead of Bayesian networks, which are another class of probabilistic graphical models, allows us to use cyclic structures. The properties of the HCI are expressed in Probabilistic Computation Tree Logic (PCTL) [9] and verified using the probabilistic model checker PRISM [10].

5 Example

In this section, we present an example that was implemented as proof of concept for our method. Due to the page limit, we only present the final model of the HCI, which is used to verify quantitative properties.

We model the interaction of a user with an email client. The user intends to write a confidential email and send it to *Alice*. However, with a certain probability, he/she performs an erroneous action choosing *Bob* instead, which results in high costs as confidential information is disclosed. Then, among others, the following erroneous interaction sequences are possible (with certain probabilities):

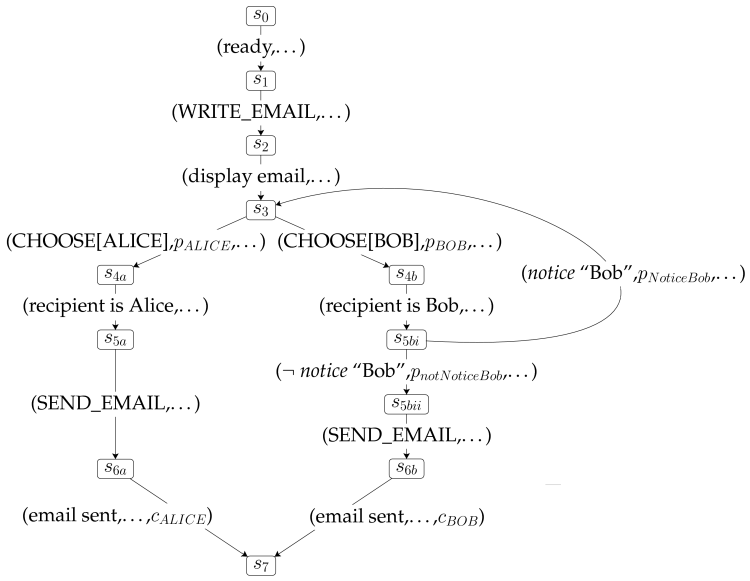

Fig. 3. Part of the PVIOLTS of the mailing example. Labels written in uppercase denote user's inputs, lowercase denote the software's outputs. The interpretation of the application's output by the user is included only for the case that *Bob* is chosen as the recipient.

- The user accidentally selects *Bob* as the addressee. The application consequently reacts with setting *Bob* to be the recipient. Based on the output provided by the application, the user's assumptions about the applications's state, in particular the addressee, are not met and the user notices the error. The user corrects the error. Then, compared to the non-erroneous situation, the total cost of the interaction is increased by the amount it takes to select and set an addressee, and to notice the mistake.
- The user accidentally selects *Bob*, and the application sets *Bob* as recipient. In contrast to the previous sequence of actions, the user interprets the application's output incorrectly, assuming that *Alice* is selected. Thus the user fails to notice the first error, and as a second error sends the email to *Bob*. Here, the error cost is the cost of sending confidential data to the wrong recipient.
- The user accidentally selects *Bob*, the application sets *Bob*, and the user notices the error based on the application's output. However, the user reacts with a second error by sending the email instead of correcting the address. Again, the error cost is the cost of sending confidential data to the wrong recipient.

Once the model of the interaction is constructed using our method, properties of the interaction can be formulated in PCTL and then checked by PRISM. Figure 3 shows part of the PVIOLTS model of the interaction. For example,

the formula $P_{\geq 0.95}[\Diamond sentTo(Alice)]$ expresses that the email is sent to *Alice* in at least 95% of the interactions. To express that "with a probability of at most 0.02, *Bob* is selected as the addressee and the error is not corrected", the property $P_{\leq 0.02}[\Diamond(chosen(Bob) \wedge (chosen(Bob)\ U\ sentTo(Bob)))]$ can be used. With PRISM's support of cost analysis, the average total cost for sending an email can be computed by the query $R =?\ [F\ email\_sent]$. Due to the possibility of erroneous interaction, the average total cost here will be higher than in a scenario where errors do not result in additional cost.

A way to lower the probability of sending the email to *Bob* would be to introduce additional dialogue boxes that require the user to confirm the address selection. If the user is modeled to react "reasonably" to a confirmation request with a high probability (as opposed to, e.g., blindly confirming the selection), it can be proven that the probability to send the confidential email to *Bob* is indeed lowered.

6 Modeling Changing Probabilities

In the probabilistic HCI models described so far, the probabilities and costs are fixed for each action. In certain situations, however, it is useful to model changing probabilities and costs. This can be done by adding a set S of variables to the states of the transition system whose values can be changed by state transitions. Probabilities and costs can then be modeled as function of the values of these state variables, which greatly improves the expressiveness of our approach. For example, the probability $p(a)$ of an action a becomes a probability $p(a, S)$. The variables in S can be flags, counters, etc. For first experiments, this concept of changing probabilities has been implemented.

Some examples for the use of this extension in the context of the email scenario from the previous section are:

- different probabilities for an action can be used for different levels of complexity of a confirmation:

$$p(a, S) = \begin{cases} 0.7 & \text{if } S.confirmationType = simple \\ 0.9 & \text{if } S.confirmationType = complex \end{cases}$$

- a "learning" user can be modeled by increasing the probability for the correct action over time:

$$p(a, S) = \begin{cases} 0.6 \quad \text{and} \quad \{S.learned := true\} & \text{if } \neg S.learned \\ 0.8 \quad \text{and} \quad \{S.learned := true\} & \text{if } S.learned \end{cases}$$

- an increasingly bored and thus "inattentive" user can modeled by decreasing this probability over time:

$$p(a, S) = \frac{1}{S.inattentionLevel + 2} + 0.4 \quad \text{and} \quad \{S.inattentionLevel{+}{+}\}$$

  yields the sequence $0.9, 0.73, 0.65, 0.6, \ldots$ if $S.inattentionLevel = 0$ initially.

Confirmations are modeled on the user side as well as the application side. While they decrease total cost of the application's actions as unintended actions are avoided, they increase total cost of the user's action as he has to interpret the confirmation request and react to it. Using our method, one can analyse the trade off between the cognitive workload that is imposed on the user by complex confirmations, and the average cost that is saved.

7 Conclusion

In this paper we have introduced a method for the formalization of probabilistic models of human-computer interaction (HCI) including user behavior. These models can then be used for the analysis and verification of HCI systems with the support of model checking tools. This allows to answer questions, such as "what is the probability that the user will send confidential information to unauthorized recipients?" and to verify the corresponding properties of the HCI model.

Our method can help to develop user interfaces by avoiding the trap of having to do human error analysis at the latest stages in the design process. Furthermore, it can help a designer to validate assumptions about human performance. By setting up several scenarios, the analyst is able to discover the impact of alternative designs on the expected cost of HCI. Thus, formal modeling and an examination of the expected costs can together contribute to the design of user interfaces, which have to be robust to the error-prone behavior of humans.

References

1. Avery, L.W., Sanquist, T.F., O'Mara, P.A., Shepard, A.P., Donohoo, D.T.: U.S. Army weapon systems human-computer interface style guide. Version 2 (2007)
2. Shneiderman, B.: Designing the User Interface. Third edn. Addison Wesley (1998)
3. Leveson, N.G.: Analyzing software specifications for mode confusion potential. In: Proc. of the Workshop on Human Error and System Development. (1997)
4. Beckert, B., Beuster, G.: A method for formalizing, analyzing, and verifying secure user interfaces. In Jifeng, H., Liu, Z., eds.: Proc., Int. Conf. on Formal Engineering Methods (ICFEM), Macao, China. LNCS 4260, Springer (2006) 55–73
5. John, B.E., Kieras, D.E.: The GOMS family of user interface analysis techniques: comparison and contrast. ACM Transactions on Computer-Human Interaction **3**(4) (December 1996) 320–351
6. Swain, A., Guttman, H.E.: Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications. Sandia National Laboratories (1983)
7. Kirwan, B.: A Guide to Practical Human Reliability Assessment. Taylor and Francis, London, UK (1994)
8. Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Proceedings, TACAS, Vienna, Austria. LNCS 3920, Springer (2006) 441–444
9. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing **6**(5) (1994) 512–535
10. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In: Int. Conf. on Computer Performance Evaluation, Modelling Techniques and Tools (TOOLS), London, UK. LNCS 2324, Springer (2002) 200–204