Probabilistic User Models for the Verification of Human-Computer Interaction

Markus Wagner

wagnermar@uni-koblenz.de

Abstract: We present a method that allows the formalization, analysis, and verification of probabilistic human-computer interaction (HCI) with the support of model checking tools. Based on a method for the formalization of HCI under security aspects, our method allows us to model probabilistic user behavior as well as to offer support for the computation of the interaction's cost. It enables us to verify quantitative aspects of the interaction and to answer questions like "what is the probability that the user will send confidential information to unauthorized individuals" and "will the user (on average) be able to send the email within a given amount of time?"

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 [ASO+07]. Unfortunately, this control does not always work perfectly and errors in HCI occur. An error can be regarded as a deviation of the achieved result from the intended result. 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 user interacted incorrectly with the computer, or someone attacked the system. The consideration of user errors and their overall impact on the system should form an important part of an analysis of a system's usability, safety, and security.

Several techniques available can be used to verify the functional correctness of a system. Examples of such techniques are theorem proving, simulation/testing and model checking. We focus on the last technique, namely formal verification by means of model checking [CGP99]. The goal of our technique is to proof properties of the interaction, or more specifically, to formally prove that all possible ways of HCI in a given scenario conform to the requirements. The method of [BB06] allows the formalization, analysis, and verification of secure user interfaces and serves as the basis of our work. It is based on GOMS [JK96], which is a well-established user modeling method. In [BB06], the formal GOMS models are augmented with formal models of the application and the user's assumptions, finally allowing formal reasoning about the security of user interfaces.

2 Erroneous Interaction

The user of an application is not always aware of all possible consequences of his/her interaction potential. While in most cases only "time" is lost due to incorrect interaction, money is lost as well, or even humans injured. To distinguish between the different effects

of actions the term of *action costs* is introduced. This term is used to take quantities like "amount of time", "amount of money" or "amount of resources" into consideration. 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 individuals. In order to undo errors, additional actions have to be carried out, introducing *error costs*. In other words, human errors add extra costs to the total sum of the sequence of actions. Based on the total interaction cost and on the idea that the user has an amount of resources he is willing to invest during the interaction, the sequences can be classified into three categories: (1) optimal sequences with minimal total costs, (2) acceptable sequences with total costs that do not exceed the budget, and (3) unacceptable sequences with total costs that exceed the user's budget.

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 [Shn98, ASO$^+$07, Lev97]. In general, most rules are obtained reflectively from experience, meaning that they are based on a combination of expert knowledge and empirical approaches to refine the rules. Furthermore, a number of techniques is known to help in making predictions about the reliability of a system when humans interact with it [SG83, Kir94]. They consist mainly of databases containing probabilities of different kinds of human errors. The error data comes from many sources, e.g. nuclear power plants, simulator studies, laboratory experiments etc. A method is provided for combing the data in order to produce estimations of the erroneous executions of tasks.

3 Formal Analysis of Erroneous Actions

In the following, we present our approach to the formalization, analysis, and verification of probabilistic HCI. Following [BB06], we assume that a user can only interact correctly with a system if he always has a correct assumption about the configuration of the application, including the internal state and relevant data. This means in terms of Linear Temporal Logic $\mathbf{G}((a_0 \leftrightarrow c_0) \wedge (a_1 \leftrightarrow c_1) \wedge \cdots \wedge (a_n \leftrightarrow c_n))$, where $a_0, \ldots, a_n$ represent the critical properties of the application, and $c_0, \ldots, c_n$ represent the user's assumption about critical properties. If a user fulfills this requirement, he is not error-prone. If he does not, the following scenarios are possible: (1) parts of the user's assumptions are wrong, and (2) parts of the user's assumptions are missing (over-abstraction).

In [BB06], *Input Output Labeled Transition Systems* (IOLTS) are used to model the components of the interaction. An IOLTS is a tupel $L = (S, \Sigma, s_0, \rightarrow)$ 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 a set of *labels* $\Sigma = \Sigma? \cup \Sigma! \cup \Sigma I$. We call $\Sigma?$ the *input alphabet*, $\Sigma!$ the *output alphabet*, and ΣI the *internal alphabet*. For example, the software's state is represented by its internal state and data. Whenever the user executes an action, the corresponding state transition is performed. The corresponding edges in the automaton are annotated with a label denoting the action.

In our approach, three components are needed in order to apply automated reasoning to HCI: (1) a formal GOMS model and its corresponding IOLTS, (2) a component representing the user's assumptions of the software's state and its corresponding IOLTS, and (3) a component representing the application itself and its corresponding IOLTS. The method

of [BB06] uses these three components to produce a complete model of the interaction, which can be used as input for a model checker.

In our expansion, we continue to model the GOMS description using an IOLTS. The application's model is extended by numerical values, representing the costs for the execution of each single step. We use a *probabilistic IOLTS with costs* to model the user's assumptions. Thus, not only the probabilistic behavior can be modeled, but also "personal" arising costs. As sources for the statistical data, we suggest databases that are mentioned in the previous section. However, this is not mandatory and the analyst can come up with his/her own values. For the component representing the application itself, an *IOLTS with costs* is used. Now, the mutual composition of these three components provides the expanded model of the HCI (see Figure 1), incorporating probabilistic user behavior as well as the interaction's costs.

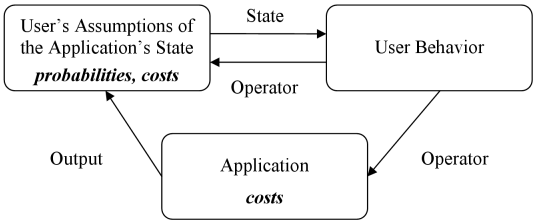

Figure 1: Extended model of the interaction.

The probabilistic model that is the output of 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 [HKNP06]. In our setting, properties of the HCI are expressed in Probabilistic Computation Tree Logic (PCTL) [HJ94] and verified using the probabilistic model checker PRISM [KNP02].

For demonstration purposes, let us think of a scenario where a user interacts with an email program. She intends to write a confidential email and send it to *Alice*. However, with a certain probability, she can choose *Bob* as well, which will result in high costs. For example, the following situations are possible: (1) the user accidentally selects *Bob* as the addressee, notices her mistake, and corrects her mistake, (2) she selects *Bob* again, but does not notice her mistake, and sends the email to *Bob*, and (3) she selects *Bob*, notices it, but fails to change the addressee, and sends the email to *Bob*. 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. For example, if the property $P_{\geq 0.95}[true\ U\ sentTo(Alice)]$ holds, it is formally proven that the email is sent to *Alice* in at least 95% of the the interactions. The property, $P_{\leq 0.20}[chosen(Bob)\ U\ sentTo(Bob)]$ can be used to express that "with a probability of at most 0.20, the selection of *Bob* as the addressee is not corrected". A way to lower the probability of sending the email to *Bob* would be to introduce additional dialogue boxes that would require the user to confirm her selection. If the user is modeled to react "reasonably" (versus "inattentively") to a confirmation, it can be proven that the probability to send the email to *Bob* is lowered. Finally, with PRISM's support of cost analysis, the average total cost for sending an email can be computed by the property $R =?\ [F\ email\_sent]$. 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.

References

[ASO+07] Larry W. Avery, Thomas F. Sanquist, Peter A. O'Mara, Anthony P. Shepard, and Daniel T. Donohoo. U.S. Army weapon systems human-computer interface style guide. Version 2, April 30 2007.

[BB06] Bernhard Beckert and Gerd Beuster. A Method for Formalizing, Analyzing, and Verifying Secure User Interfaces. In *ICFEM*, volume 4260 of *Lecture Notes in Computer Science*, pages 55–73. Springer, 2006.

[CGP99] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[HJ94] Hans Hansson and Bengt Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[HKNP06] Andrew Hinton, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer, 2006.

[JK96] Bonnie E. John and David E. Kieras. The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320–351, December 1996.

[Kir94] Barry Kirwan. *A Guide to Practical Human Reliability Assessment*. Taylor and Francis, London, UK, 1994.

[KNP02] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: Probabilistic Symbolic Model Checker. *Lecture Notes in Computer Science*, 2324:200–204, 2002.

[Lev97] Nancy G. Leveson. Analyzing Software Specifications for Mode Confusion Potential. In *Proc. of the Workshop on Human Error and System Development*. 1997.

[SG83] Alan Swain and H. E. Guttman. *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications*. Sandia National Laboratories, 1983.

[Shn98] Ben Shneiderman. *Designing the User Interface*. Addison Wesley Longman, third edition, 1998.