Electronic Communications of the EASST
No \volume defined!

Operational Model: Integrating User Tasks and Environment Information with System Model

4 pages

No *ed(s) defined!
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Operational Model: Integrating User Tasks and Environment Information with System Model

Abstract: This paper addresses the problem of integrating information about user tasks and about the operating environment to the model of a system. Following a modelling based on labelled transition systems, this integration can be done with elementary operations: models synchronization and graph operations. Integration of user tasks and information about operating environment allows to get operational model which represents the knowledge the user should have about the system to perform a set of tasks, given information provided by the operating environment, through user interface for example. The paper draws up a formal way to do the integration to get an operational model which can be used to evaluate and compare different system's design, to do verification or to generate training material.

Keywords: Formal methods, Human-Computer Interaction, Parallel synchronization, User tasks, System Design Evaluation and Comparison

1 Introduction

When analyzing and verifying the design and the specification of a system, besides the system itself, the integration of user tasks plays an important role. The user tasks which represents what the user wants to do with the system — his goals — can change completely the way a system is conceived and designed. There are a lot of whole community using formal methods to analyse Human-Computer Interaction (HCI), providing rigorous, systematic and automatable way of analysis. Information about user tasks is used to help the analysis of specification [Cam03, PS01] or to analyse the effects of erroneous human behaviour by building deviations in how tasks are performed by the user [PS02, BB07].

This paper proposes to integrate informations about user tasks and operating environment with the description of the system. This integration provides an *operational model* of the system which represents part of the full behaviour of the system that is relevant according to the user tasks and the operation environment.

2 Operational Model

The *operational model* of a system only covers the behaviour part of the system model which is relevant for the user. Indeed, the user does not need to know the full behaviour of the system to

use it because he only wants to perform some *tasks* with it. Moreover, when operating the system, the user gets some feedback from it, that is precisely what we call the *operating environment*. That feedback may help him to operate the system. Figure 1 illustrates the components the operational model depends on: the *system model* represents the full behaviour of the system, the *tasks model* represents the tasks that the user should be able to perform on the system and the *action-based user interface* represents part of the user interface, that is information about which event is occuring in the system.

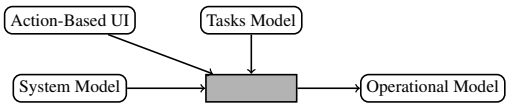

Figure 1: Computing operational model from system and tasks models.

The system is modelled as a *labelled transition system* (LTS) $\mathscr{M} = \langle S, \mathscr{L}, s_0, \rightarrow \rangle$ with S the set of states, $\mathscr{L}$ the set of actions, s_0 the initial state and $\rightarrow \subseteq S \times \mathscr{L} \times S$ the transition relation. The τ action denotes an *internal action* representing all transitions that are not observable from outside the system. The *action-based user interface* consists in a distinction among observable actions: *commands* are performed by the user on the system and *observations* are controlled by the system and occur autonomously without any user action.

2.1 Integrating Tasks

User tasks are modelled using ConcurTaskTrees [PMM97] which is a graphical notation that allows designers to describe hierarchies of tasks linked with temporal relations following the semantic of LOTOS [ISO89]. To fit in our framework, all the user and application tasks that are leaves should come from $\mathscr{L}$, user tasks corresponding to commands and application tasks to observations. To be able to integrate user tasks with the system model, they are transformed from CTT descriptions to a set of LTSs $\{\mathscr{T}_1, \cdots, \mathscr{T}_n\}$ which can be done with the tool developped in [PS01] for example.

The integration of the user tasks with the system is done with a *synchronous parallel composition* between the system model $\mathscr{M}$ and the tasks model $\mathscr{T}$, synchronized on $\mathscr{L} \setminus \{\tau\}$. The tasks model $\mathscr{T}$ is built from the set $\{\mathscr{T}_1, \cdots, \mathscr{T}_n\}$ of user tasks, adding transitions $s_{0_\mathscr{T}} \xrightarrow{\tau} \mathscr{T}_i$ for each user task $\mathscr{T}_i$, where $s_{0_\mathscr{T}}$ is the initial state of $\mathscr{T}$. The model $\mathscr{M} \parallel \mathscr{T}$ can contains internal transitions which are not relevant for the user; they are removed using the *edge contraction* operation from graph theory.

2.2 Integrating Environment Information

Information about the operationg environment can be integrated by performing two simplifications on the system model. The first simplification that can be done comes from the idea that the user is not obliged to know all the paths of the system exactly. Some paths can indeed be ignored by the user provided that the user knows that the path has been followed, for example through the user interface. The second simplification comes from the hierarchy in the user tasks; the user

should is not required to know the system at the finer level, some actions can be gathered into one single action.

Figure 2 illustrates the two simplification. The system modelled is a vending machine accepting credit card or cash. If the user selects creditCard, after encoding his/her PIN number, the system will enter a procedure to contact the bank. The progress is shown to the user through the user interface and he/she can so follow the transaction thanks to the observations. All the states from the one preceding the connected action to the one after the dataRecv action can be merged into a single state, preserving any edges linked to any of the merged states. The second kind of simplification can be done according to the tasks hierarchy. Here, the actions 5in and 10in refine addMoney. All these actions can be renamed and states linked with the addMoney action can be merged because the user does not need to distinguish the 5in and 10in anymore.

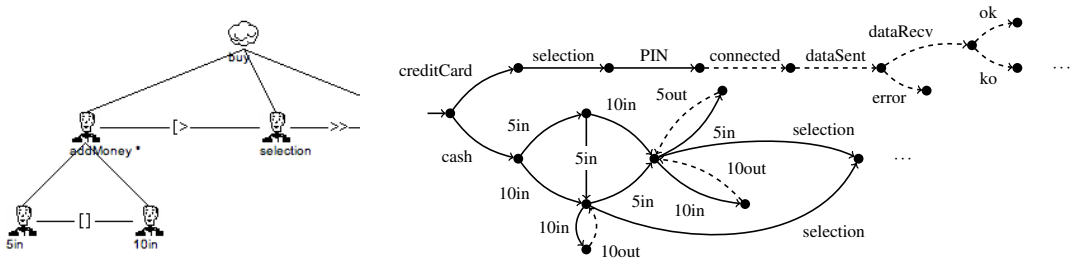

Figure 2: Example of a user task (left) and of a system model illustrating the integration of environment information (right). The system is a vending machine accepting payment with credit card or cash. Plain lines represent commands and dashed lines represent observations.

2.3 Discussion

The operational model of a system captures exactly the behaviour needed by the user to perform a set of tasks given an operating environment. This model can then be used for several purpose. It can be used to generate training manual or to verify whether existing manual's content concurs with the knowledge needed for the user. Viewing a manual as a set of scenarios, it is possible to generate a corresponding model [DLDL05] and then to compare it with the operational model. Operational model can also be used to compare different designs for a system given a set of tasks to be performed on the system. Operational models can be computed for each design and then compared, for example using some metrics which can range from simpler one like the number of states to more complex like the number of different paths that can be used to perform each task.

3 Conclusion and Perspectives

This paper defines the notion of *operational model* of a system. An operational model captures the part of the behaviour of a system which is relevant to a user who should be able to perform some tasks and gets information about the operation environment through the user interface. Such a model can be used for different purpose: generation of training user manual, evaluation and comparison of different systems, checking properties, ...

Further work includes implementing a prototype for the computation of an operational model for a given system. Another direction consists in using operational model to evaluate and compare system models based on a set of user tasks, assessing the usability of the system using a metric on the operational model.

Acknowledgements: This work is partly supported by project MoVES under the Interuniversity Attraction Poles Programme — Belgian State — Belgian Science Policy.

Bibliography

[BB07] R. Bastide, S. Basnyat. Error Patterns: Systematic Investifation of Deviations in Task Models. In Coninx et al. (eds.), *Proceedings of the 5th International Workshop on Task Models Diagrams for UI Design*. Lecture Notes in Computer Science 4285, pp. 109–121. Springer-Verlag, 2007.

[Cam03] J. C. Campos. Using task knowledge to guide interactor specifications analysis. In *Proceedings of the 10th International Workshop on Design, Specification and Verification of Interactive Systems*. Lecture Notes in Computer Science 2844, pp. 171–186. Springer-Verlag, 2003.

[DLDL05] C. Damas, B. Lambeau, P. Dupont, A. van Lamsweerde. Generating Annotated Behavior Models from End-User Scenarios. *IEEE Transactions on Software Engineering* 31(12):1056–1073, Dec. 2005.

[ISO89] ISO 8807:1989. *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*. International Organization for Standardization, Geneva, Switzerland, 1989.

[PMM97] F. Paternò, C. Mancini, S. Meniconi. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*. Pp. 362–369. Chapman & Hall, Ltd., London, UK, 1997.

[PS01] F. Paternò, C. Santoro. Integrating Model Checking and HCI Tools to Help Designers Verify User Interface Properties. In *Proceedings of the 7th International Workshop on Design, Specification and Verification of Interactive Systems*. Lecture Notes in Computer Science 1946, pp. 135–150. Springer-Verlag, 2001.

[PS02] F. Paternò, C. Santoro. Preventing User Errors by Systematic Analysis of Deviations from the System Task Model. *International Journal of Human-Computer Studies* 56(2):225–245, Feb. 2002.