Efficient Modelling and Analysis of User Interfaces in High-Assurance Systems

Saulo Rodrigues e Silva[⋆]

HASLab/INESC TEC & Universidade do Minho
Campus de Gualtar, Braga, Portugal
saulo.r.silva@inesctec.pt

Abstract. This paper presents a research strand on the convergence between models of device interaction logic and models of user goals and activities. The main objective is to improve the quality of modelling and analysis of high-assurance interactive system's interfaces by exploring the integration of the two types of models.

Keywords: Interactive Human-Machine Systems; Task Analysis; Formal Verification.

1 Introduction and motivation

Cyber-Physical Systems (CPS) integrate computational and physical capabilities, and have, in many cases, high-assurance needs. Interactive CPS such as cockpits and medical devices provide user interfaces that allow users to monitor and control the system. To ensure safe and effective operation of these interactive CPS, it is important to ensure the absence of latent design anomalies in their user interfaces.

Three main types of approaches exist for formal modelling and analysis of human-machine interaction, each tackling the verification of user interface design from a different and complementary perspective:

– Analysis of usability and safety properties of user interface design. This approach aims to verify that the behaviour of the user interface is compliant with properties capturing best practice in human-machine interface design. An example property is *visibility of operational modes*, which aims to ensure that the user interface presents sufficient information about the current operational mode of the device. Campos and Harrison [1] is an example of such approach. A typical challenge with these approaches is the scalability of the analysis, as all possible system behaviours and all possible user interactions with the system need to be considered. Another challenge is also the plausibility or relevance of the user behaviours produced by the analysis, as the obtained counter examples can contain *random* or *unrealistic* human-machine interactions.

[⋆] Copyright held by the author.

- Analysis of user interface design against task models. Task models capture sequences of actions the user needs to carry out to interact with the system when achieving a goal. These sequences of actions are typically described using a hierarchical decomposition of goals into subgoals and atomic user actions and system events. This approach aims to check that the user interface correctly supports tasks representing operations described, e.g., in training material, and best/actual practice. Examples approaches include Palanque's work [2], Bolton's work [3], Paternò's work [4] and Campos's work [5]. A challenge related to these approaches is how to analyse systematically non-normative behaviours followed by users, e.g., in abnormal system conditions, or to take into account strategies adopted by the user to optimise task operations [3].
- Analysis of user interface design against human behaviour. This approach involves explicit definition of user models capturing cognitive assumptions about the decision-making process followed by the users when operating an interactive system. Example approaches include Rushby's work on automation surprise [6], Rimvydas's *et. al.* work on generic user models [7] and Degani's work on mental models [8]. A challenge with these approaches is how to validate the cognitive assumptions incorporated in the user model.

Efficient methods for developing models and carrying out analysis of user interface design are key to make these methods acceptable to industry. While there is on going work on defining modelling patterns and analysis templates (e.g., see Harrison *et. al.* [9], and Bowen and Reeves' work [10]), little work has been done on exploring how to improve efficiency of the approaches by combining them.

This research aims to address this challenge. We are initially focusing on exploring how to combine two types of analysis: verification against usability and safety properties, and verification against task models. The expected outcome is a set of design patterns presenting efficient solutions to combine these two approaches.

2 Approaches combining task analysis and device analysis

We now highlight some of the most recent approaches combining task analysis with user interface design analysis.

Campos [5] describes the system model and the task model as interactors. An interactor is an object with state and operations. As users can perceive the interactor's state and access its operations, it provides a way to write interactive system's behavioural properties. The behaviour of the interator is expressed in Modal Action Logic (MAL). The properties to be proved are expressed in CTL. Analysis is performed with the IVY tool, which enables the automatic translation of interactors models into nuSMV [11] models and properties.

Paternò [4] makes a distinction between three types of task models, each having a different role in the analysis of a user interface design. One type of task model defines the *task model of the existing system*. This model describes how tasks should be carried out in the current system, according to the actual implementation of the system. Another task model is the *task model of an envisioned system*, which is used at the early stages of system development, to shape the functionalities of the human-machine interface of the system under development. Finally, the *user task model* captures hypotheses about how the user thinks a task should be accomplished with the system.

Palanque *et. al.* [12] use the HAMSTERS notation to describe task models, and Interactive Cooperative Objects (ICO) [13], a Petri Nets based language, to describe the system model. They establish a correspondence between actions in the task model and events described in the system model. Using this correspondence they can co-execute task models and system models. They analyse the system model against specific sequences of actions presented in the task model. This process is supported by the CIRCUS toolset.

Campos *et. al.* [14] extend Palanque's approach by automatically generating scenarios that can be used to test system models against task models. This extension is used for automated exploration of test scenarios representing normative behaviour, as well as for exploration of possible use errors and deviations from normative behaviour.

Bolton *et. al.* [15] combine task models and system models based on the idea of automatically translating task models into Temporal Logic Specifications (TLS). Task models are represented using the Extended Operator Function Model (EOFM) notation. The EOFM notation is a task model representation based on the XML language. The system model is described as a state machine in SAL [16]. Compliance between system model and task model is analysed in SAL, by checking that the system model satisfies the TLS properties. A challenge with this approach is the scalability of the analysis for realistic systems. Solutions to address this concern are being explored in [17].

3 Tools for analysis of user interface design and tasks

In the first months of this research, we have investigated different tools for modelling systems and user tasks. We are currently focusing our attention on PVSio-web [18] and HAMSTERS [19], each providing a different formal modelling approach and formal analysis technique.

We consider the PVSio-web toolkit for modelling the interactive behaviour of the system due to its theorem proving capabilities. Although the analysis is not automatic and proving properties may require human intervention, theorem proving does not suffer state space explosion issues faced by model-checking approaches, nor issues with incompleteness of the analysis faced by simulation-based approaches.

We consider HAMSTERS due to the possibility of translating task models into state machines [14]. This capability of the tool allows us to express task models in a language compatible with that used for modelling the system behaviour.

3.1 Modelling the system behaviour in PVSio-web

PVSio-web [18] is a prototyping and analysis toolkit based on the PVS [20] verification system. The functionalities of PVSio-web are similar to those of commercial tool suites, such as MathWorks Simulink[1], SCADE[2] and IBM's Rational Statemate[3]. The tool offers a graphical environment to define the visual appearance of the prototype user interface, as well as the interactive behaviour of the prototype. The visual aspect is based on a picture of the device. The behaviour is a PVS executable model. The PVS model can be developed using a graphical notation, Emucharts, which is a dialect of Statecharts. The Emucharts notation supports *states*, representing the different modes of the system, *state variables*, representing the structure of the system state, and *transitions*, representing events that change the system state. The semantics of Emucharts is formally defined in the PVS higher-order logic language.

3.2 Modelling tasks with HAMSTERS

HAMSTERS is both a tool and a notation that allows creating, editing and simulating task models. It has a graphical and hierarchical notation to represent human activities, based on Concur Task Trees (CTT) [4]. HAMSTERS supports simulation based analysis of task models, as well as quantitative analysis of cognitive workload based on the type of human machine interactions required in the task. HAMSTERS is included in the CIRCUS toolset, which also includes PetShop (for modelling the system behaviour) and SWAN (for co-execution of task models and system models). Recently, Martinie *et. al.* [19] extended the tool capabilities with extensions to the task model notation suitable to describe activities involving multiple users.

4 Conclusion

The work presented here summarises the current state of my research on integrating system modelling and analysis with task modelling and analysis. The work is at the early stages. Two formal tools are currently being used that support well known notations that can be translated into state machines. This makes it easier to investigate the definition of efficient modelling patterns combining tasks models and systems models. Future work includes moving to a realistic case study, in the medical or avionics domain. This case study will provide us with a test-bench suitable to inform and validate the development of the modelling patterns.

Acknowledgement. This project is partially supported by Project "NORTE-01-0145-FEDER-000016", financed by the North Portugal Regional Operational

[1] `http://uk.mathworks.com/products/simulink/`
[2] `http://www.esterel-technologies.com/`
[3] `http://www-03.ibm.com/software/products/en/ratistat`

Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF). It is also supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) PhD scholarship.

References

1. J. C. Campos and M. D. Harrison, "Systematic analysis of control panel interfaces using formal tools," in *Interactive systems. Design, specification, and verification*, pp. 72–85, Springer, 2008.
2. P. Palanque, R. Bastide, and V. Sengès, "Validating interactive system design through the verification of formal task and system models," in *Engineering for Human-Computer Interaction*, pp. 189–212, Springer, 1996.
3. M. L. Bolton, R. I. Siminiceanu, and E. J. Bass, "A systematic approach to model checking human–automation interaction using task analytic models," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 5, pp. 961–976, 2011.
4. F. Paternò, "ConcurTaskTrees: an engineered notation for task models," *The handbook of task analysis for human-computer interaction*, pp. 483–503, 2004.
5. J. C. Campos, "Using task knowledge to guide interactor specifications analysis," in *International Workshop on Design, Specification, and Verification of Interactive Systems*, pp. 171–186, Springer, 2003.
6. J. Rushby, "Using model checking to help discover mode confusions and other automation surprises," *Reliability Engineering & System Safety*, vol. 75, no. 2, pp. 167–177, 2002.
7. P. Curzon, R. Rukšėnas, and A. Blandford, "An approach to formal verification of human–computer interaction," *Formal Aspects of Computing*, vol. 19, no. 4, pp. 513–550, 2007.
8. A. Degani, *Taming HAL: Designing interfaces beyond 2001.* Springer, 2004.
9. M. D. Harrison, J. C. Campos, R. Rukšėnas, and P. Curzon, "Modelling Information Resources and Their Salience in Medical Device Design," in *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '16, (New York, NY, USA), pp. 194–203, ACM, 2016.
10. J. Bowen and S. Reeves, "Design Patterns for Models of Interactive Systems," in *Software Engineering Conference (ASWEC), 24th Australasian*, pp. 223–232, IEEE, 2015.
11. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking," in *International Conference on Computer Aided Verification*, pp. 359–364, Springer, 2002.
12. E. Barboni, J.-F. Ladry, D. Navarre, P. Palanque, and M. Winckler, "Beyond Modelling: An Integrated Environment Supporting Co-execution of Tasks and Systems Models," in *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '10, (New York, NY, USA), pp. 165–174, ACM, 2010.
13. P. Palanque, M. Winckler, and C. Martinie, "A formal model-based approach for designing interruptions-tolerant advanced user interfaces," in *Model-Driven Development of Advanced User Interfaces*, pp. 143–169, Springer, 2011.

14. J. C. Campos, C. Fayollas, C. Martinie, D. Navarre, P. Palanque, and M. Pinto, "Systematic Automation of Scenario-based Testing of User Interfaces," in *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '16, (New York, NY, USA), pp. 138–148, ACM, 2016.
15. M. L. Bolton, "Automatic validation and failure diagnosis of human-device interfaces using task analytic models and model checking," *Computational and Mathematical Organization Theory*, vol. 19, no. 3, pp. 288–312, 2013.
16. L. De Moura, S. Owre, and N. Shankar, "The SAL language manual (Tech. Rep. No. CSL-01-01)," *Menlo Park: Computer Science Laboratory, SRI International*, 2003.
17. M. L. Bolton, X. Zheng, K. Molinaro, A. Houser, and M. Li, "Improving the scalability of formal human–automation interaction verification analyses that use task-analytic models," *Innovations in Systems and Software Engineering*, pp. 1–17, 2016.
18. P. Masci, P. Oladimeji, Y. Zhang, P. Jones, P. Curzon, and H. Thimbleby, "PVSio-web 2.0: Joining PVS to HCI," in *International Conference on Computer Aided Verification*, pp. 470–478, Springer, 2015.
19. C. Martinie, P. Palanque, E. Barboni, M. Winckler, M. Ragosta, A. Pasquini, and P. Lanzi, "Formal tasks and systems models as a tool for specifying and assessing automation designs," in *Proceedings of the 1st International Conference on Application and Theory of Automation in Command and Control Systems*, pp. 50–59, IRIT Press, 2011.
20. S. Owre, J. M. Rushby, and N. Shankar, "PVS: A prototype verification system," in *International Conference on Automated Deduction*, pp. 748–752, Springer, 1992.