Formal Verification of Interactive Computing Systems: Opportunities and Challenges

José C. Campos[1][0000−0001−9163−580X] and Michael D. Harrison[2][0000−0002−5567−9650]

[1] Department of Informatics/University of Minho & HASLab/INESC TEC
Braga, Portugal
jose.campos@di.uminho.pt
http://www.di.uminho.pt/~jfc
[2] School of Computing, Newcastle University
Newcastle upon Tyne, UK
michael.harrison@ncl.ac.uk
http://www.ncl.ac.uk/computing/people/profile/michael.harrison

Abstract. Formal verification has the potential to provide a level of evidence based assurance not possible by more traditional development approaches. For this potential to be fulfilled, its integration into existing practices must be achieved. Starting from this premise, the position paper discusses the opportunities created and the challenges faced by the use of formal verification in the analysis of critical interactive computing systems. Three main challenges are discussed: the accessibility of the modelling stage; support for expressing relevant properties; the need to provide analysis results that are comprehensible to a broad range of expertise including software, safety and human factors.

Keywords: Formal verification · Automated reasoning tools · Interactive computing systems.

1 Introduction

Safety and mission critical interactive computing systems require a level of evidence based assurance that traditional user centred approaches alone cannot provide. For this reason user centred approaches must be integrated with hazard analysis and risk assessment approaches to guarantee safety. Such processes are required to comply with regulatory requirements. Current approaches to provide evidence for safety, are typically based on inspection and testing techniques making it hard to guarantee an adequate level of safety assurance at a reasonable cost/effort level.

Formal verification tools promise the scaleable analysis of components of real systems including interactive systems. They enable exhaustive analysis of use centred safety requirements as part of a risk analysis. This position paper discusses extensions to existing tools for formal modelling and analysis that have the potential to enable their use by current development teams who are not expert

in formal methods. In doing this, it discusses requirements for formal toolsets that would aid their acceptability by development teams. The requirements are discussed using the IVY toolset as an example.

2 Formal verification of interactive computing systems

Formal verification, applied to interactive computing systems, has seen considerable development, mostly at the model-based level[3] [8, 4, 19, 1, 21]. In [12] we first argued that formal verification has a role to play in systematic usability analysis. Formal verification techniques, although narrower in scope, provide a more thorough analysis in which human factors claims are more clearly identified and substantiated. Use centred requirements may be identified informally by domain or human factors experts and formulated as properties that may be proved of a formal model of the interactive system under investigation. There is potential for using the complementary expertise of multiple parties to the design process. Since the publication of that paper, tools have continued to mature and their role in the context of user centred design has become more feasible.

Tools such as the IVY workbench have been shown to be applicable to real systems [3] and to contribute to the risk analysis of actual medical devices [10]. IVY focuses on model-based analysis of interactive computing system designs, focussing particularly on aspects related to their behaviour. Other tools also aim at supporting the analysis of these systems, each with its particular focus (see [6] for a comparison of CIRCUS and PVSio-web).

These activities are consistent with recent progress in "lightweight formal methods" (LFM). According to Zamansky et al.'s review [22] a key feature of developments in lightweight techniques is a focus on partial models and analyses — the ability to use formal methods to model and analyse components of the software, for example the control component, or in the present context, the user interface component. Analyses can then contribute to parts of the required analysis or program development. The review [22] uses a classification framework as a basis for assessment of relevant papers.

when: at which development stage should formal methods be applied?
what: for what parts/aspects of the system should formal methods be applied?
how: how rigorous shall the modelling and analysis be and what languages and tools should one use to achieve that?
who: which human resources should be deployed?

Hence, for example Osaiweran et al. [18] describe a Philips based medical project using *Analytical Software Design* (ASD). The modelling notation uses transition tables to simplify the expression of the design model. They focus on a control component where decisions depend on incoming events and not on the data parameters of these events. The described method automatically checks a set of

[3] Perhaps because this level plays well with the iterative nature of user centred approaches.

predefined properties of the model using a model checker and code is generated from the model. Our interest is to provide techniques that share characteristics with the reviewed techniques. The focus here is interactive systems, allowing the analyst to choose properties based on user centred engineering requirements and to check whether the property is true of the developed model.

3 Challenges and opportunities

The focus of this discussion is the analysis of safety issues associated with user interaction with software devices. The main problem with the use of formal techniques is that there can be a substantial learning curve associated with their use. Software engineers and human factors specialists have not been trained to use them. This lack of expertise is particularly significant when it is considered that many safety critical systems are developed by small teams of innovators. This is certainly the case, for example, in medical device development. Medical devices are often safety critical. They are often innovative and developed by small teams attached to hospitals where the drivers for the design have a medical background. They are not software engineers (or programmers) or HCI experts. The problem of safety analysis of these devices presents important challenges. Safety is usually seen (quite reasonably) in terms of clinical trials where the key focus is the clinical advantage of a functional medical device.

While it can be argued that formal verification has proved its worth in a number of practical applications, the challenge now is to make the tools available to a wider audience. How to make these techniques accessible to the small teams that develop these safety critical devices. This is the goal of the LFM community. The concern here is to make the models and analysis tools of formal methods accessible to a broader community. The experience of using the IVY tool to support the safety analysis of a neonatal dialysis machine [9, 10] indicates that the following stages are areas where broad interdisciplinary comprehension is necessary:

1. making the modelling stage accessible;
2. supporting the expression of relevant properties for verification;
3. providing analysis results in ways that are understandable and useful.

These stages mirror the concerns of LFM but the stress here is recognising the mixed community (software and safety engineering and human factors) that will be required to understand the scope and consequences of the models and analyses.

Each of these topics will now be addressed.

3.1 Making the modelling stage accessible to developers

The modelling stage must be made more accessible to non-formal methods experts. Building formal models of interactive systems is not the typical approach

to developing interactive computing systems, which often relies on the development of prototypes. There is, then, a gap to be filled between the practices of developers and the models needed for analysis. This must be filled by looking at what the current practices are, and how whatever design and development artefacts are used might be fed into the verification process (see [2] for an example).

One approach to make this accessible to developers is to establish common patterns for the architectures of components of interactive systems, to provide a framework that can be populated in the case of the particular system. This process produced the model that formed the basis for the controller in the case of the neonatal dialyser. The developers had already produced a spreadsheet that represented the state transition model that formed the basis for analysis. The spreadsheet was also used in the implementation of the controller for the device. The modelling problem then becomes one of choosing the architecture and populating it.

Several existing notations provide starting points for modellers, see for example [17]. The details of the models as instantiated for the particular problem also need to be clear to the team. Simple descriptions of state transition models have been expressed in ASD [18] as well as in much earlier approaches, including those developed by Heitmeyer's team [13], which can be useful here.

3.2 Supporting the expression of relevant properties for verification

The process of creating properties for verification is particularly challenging and one where the integration with existing practices is both more critical and more likely to create added value. The risk analysis process is often based on a set of informal requirements. Sometimes these requirements are based on previous cases, as risk logs. The risk requirements specified in the risk log are designed to mitigate hazards and may have software, hardware or human aspects to them. A safety requirement may demand an operating procedure or it may suggest a formal requirement that must be true of a software component, for example. This latter possibility could lead to formal analysis of the model of the system. The problem for the safety analysis then is to decide which aspects of a requirement can be captured by properties of a formal model and to provide a formal expression of the property.

Support for specifying these requirements is currently provided by the adoption of property specification templates [7, 11]. To ease their application, these templates must be adjusted to be relevant to the risk logs. Ideally this will enable properties for verification to be more directly drawn from the development process itself. However, our experience of the dialysis machine is that the developers produced "pseudo formal" descriptions of the requirements. The formal modeller, who was engaged in the analysis, then produced a CTL property that could be checked of the model. Another important part of this process was to allow the analyst to "go back", taking the property as formulated and showing that it actually implements the intended part of the risk requirement. While there are likely to be many requirements that fit standard templates, some require-

ments may not obviously fit and therefore the translation of "pseudo formal" expressions is likely to be necessary.

A process such as the above will be made easier through some degree of automation. Tool support is needed to go from risks, described in the logs, to properties for verification that capture the risk being considered (natural language processing techniques might be useful here [20]). Finally, work is also needed in folding considerations about use into currently available hazard analysis techniques, so that the human aspects of risks are adequately dealt with when risk logs are produced (see, for example, [16]).

3.3 Providing analysis results in ways that are understandable by, and useful to, developers

When verification fails, the causes of failure need to be identified and investigated. The counter-examples produced by model checkers can be particularly useful here, but need to be presented in an understandable manner.

Several alternatives can, and have been, explored to this end. Tools such as IVY provide a number of graphical representations to that purpose. These however assume a technical nature (tabular or activity diagram based representations of the states of the system in the counter example) that might not necessarily be the most appropriate for non-experts. At the other end of the spectrum, tools such as PVSio-web [15] support the prototyping of the user interfaces, based on their formal models. Using these prototypes to illustrate the counter examples will allow a more design oriented representation of the counter-examples.

A mid-way, more flexible approach, is explored in [5] in the context of representing the outcome of analysis with Alloy [14]. The idea is to use *managers* to support the configuration of how states and transitions between states should be graphically rendered. This provides the flexibility to explore the representation of states in different ways. While the work does not directly address interactive computing systems models, the techniques used can clearly be adjusted to support them.

4 Conclusion

Formal verification can provide a level of evidence based assurance that more traditional development approaches do not guarantee. In this paper we have discussed opportunities and challenges of its use with interactive computing systems. A common thread between the challenges is that the formal analysis process must be integrated into existing practices, being driven by them and not forcing developers to change their current practices. Areas where broad interdisciplinary comprehension is necessary to achieve this integration have been identified and briefly discussed.

Acknowledgments

This work is financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project POCI-01-0145-FEDER-016826.

References

1. Bolton, M.L., Bass, E., Siminiceanu, R.: Using formal verification to evaluate human-automation interaction: A review. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans **43**(3), 488–503 (May 2013)
2. Bowen, J., Reeves, S.: Formal models for user interface design artefacts. Innovations in Systems and Software Engineering **4**(2), 125–141 (Jun 2008). https://doi.org/10.1007/s11334-008-0049-0, `https://doi.org/10.1007/s11334-008-0049-0`
3. Campos, J., Sousa, M., Alves, M., Harrison, M.: Formal verification of a space system's user interface with the IVY workbench. IEEE Transactions on Human-Machine Systems **46**(2), 303–316 (2016). https://doi.org/10.1109/THMS.2015.2421511
4. Campos, J.C., Harrison, M.D.: Formally verifying interactive systems: A review. In: Harrison, M.D., Torres, J.C. (eds.) Design, Specification and Verification of Interactive Systems '97, pp. 109–124. Springer Computer Science, Springer-Verlag/Wien (June 1997)
5. Couto, R., Campos, J., Macedo, N., Cunha, A.: Improving the visualization of alloy instances. In: Integrated Development Environment 2018 (F-IDE 2018). Electronic Proceedings in Theoretical Computer Science, vol. 284, pp. 37–52 (2018). https://doi.org/10.4204/EPTCS.284.4
6. Fayollas, C., Martinie, C., Palanque, P., Masci, P., Harrison, M., Campos, J., Silva, S.: Evaluation of formal IDEs for human-machine interface design and analysis: the case of CIRCUS and PVSio-web. In: Proceedings of the Third Workshop on Formal Integrated Development Environment. Electronic Proceedings in Theoretical Computer Science, vol. 240, pp. 1–19 (2017). https://doi.org/10.4204/EPTCS.240.1
7. Harrison, M., Campos, J., Masci, P., Curzon, P.: Templates as heuristics for proving properties of medical devices. In: 5th EAI International Conference on Wireless Mobile Communication and Healthcare - "Transforming healthcare through innovations in mobile and wireless technologies". ACM (2015). https://doi.org/10.4108/eai.14-10-2015.2261743
8. Harrison, M., Thimbleby, H. (eds.): Formal Methods in Human-Computer Interaction. Cambridge Series on Human-Computer Interaction, Cambridge University Press (1990)
9. Harrison, M., Drinnan, M., Campos, J., Masci, P., Freitas, L., di Maria, C., Whitaker, M.: Safety analysis of software components of a dialysis machine using model checking. In: Formal Aspects of Component Software. Lecture Notes in Computer Science, vol. 10487, pp. 137–154. Springer (2017). https://doi.org/10.1007/978-3-319-68034-7_8

10. Harrison, M., Freitas, L., Drinnan, M., Campos, J., Masci, P., di Maria, C., Whitaker, M.: Formal techniques in the safety analysis of software components of a new dialysis machine. Science of Computer Programming **175**, 17–34 (April 2019). https://doi.org/10.1016/j.scico.2019.02.003
11. Harrison, M., Masci, P., Campos, J.: Verification templates for the analysis of user interface software design. IEEE Transactions on Software Engineering (accepted). https://doi.org/10.1109/TSE.2018.2804939
12. Harrison, M.D., Campos, J.C., Loer, K.: Formal analysis of interactive systems: opportunities and weaknesses. In: Cairns, P., Cox, A. (eds.) Research Methods for Human Computer Interaction, chap. 5, pp. 88–111. Cambridge University Press (2008)
13. Heitmeyer, C., Kirby, J., Labaw, B., Bharadwaj, R.: SCR: A toolset for specifying and analyzing software requirements. In: Computer Aided Verification. pp. 526–531. Springer (1998)
14. Jackson, D.: Software Abstractions. MIT Press, revised edn. (2011)
15. Masci, P., Oladimeji, P., Zhang, Y., Jones, P., Curzon, P., Thimbleby, H.: PVSio-web 2.0: Joining PVS to HCI. In: Computer Aided Verification. Lecture Notes in Computer Science, vol. 9206, pp. 470–478. Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_30
16. Masci, P., Zhang, Y., Jones, P., Campos, J.: A hazard analysis method for systematic identification of safety requirements for user interface software in medical devices. In: Software Engineering and Formal Methods. Lecture Notes in Computer Science, vol. 10469, pp. 284–299. Springer (2017)
17. Mavridou, A., Stachtiari, E., Bliudze, S., Ivanov, A., Katsaros, P., Sifakis, J.: Architecture-based design: A satellite on-board software case study. In: Kouchnarenko, O., Khosravi, R. (eds.) Formal Aspects of Component Software. Lecture Notes in Computer Science, vol. 10231, pp. 260–279. Springer (2017). https://doi.org/10.1007/978-3-319-57666-4_16
18. Osaiweran, A., Schuts, M., Hooman, J., Groote, J.F., van Rijnsoever, B.: Evaluating the effect of a lightweight formal technique in industry. International Journal on Software Tools for Technology Transfer **18**(1), 93–108 (Feb 2016). https://doi.org/10.1007/s10009-015-0374-1, `https://doi.org/10.1007/s10009-015-0374-1`
19. Palanque, P., Paternò, F. (eds.): Formal Methods in Human-Computer Interaction. Formal Approaches to Computing and Information Technology series, Springer-Verlag, London (1998)
20. Vadera, S., Meziane, F.: From English to formal specifications. The Computer Journal **37**(9), 753–763 (1994)
21. Weyers, B., Bowen, J., Dix, A., Palanque, P. (eds.): The Handbook of Formal Methods in Human-Computer Interaction. Human–Computer Interaction Series, Springer (2017)
22. Zamansky, A., Spichkova, M., Rodríguez-Navas, G., Herrmann, P., Blech, J.O.: Towards classification of lightweight formal methods. In: Damiani, E., Spanoudakis, G., Maciaszek, L. (eds.) Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018). pp. 305–313 (2018)