Early identification of software causes of use-related hazards in medical devices

[Invited Paper]

Paolo Masci
Queen Mary University of London, United Kingdom
p.m.masci@qmul.ac.uk

Paul Curzon
Queen Mary University of London, United Kingdom
p.curzon@qmul.ac.uk

Harold Thimbleby
Swansea University
United Kingdom
harold@thimbleby.net

ABSTRACT

A hazard is a potential source of physical injury or damage to people or environment, and a hazard analysis is the process of identifying all known and foreseeable hazards and their causes in a system. In this paper, we illustrate our ongoing work in collaboration with the FDA on defining a hazard analysis technique for early identification of the causes in user interface software design of use-related hazards. The technique integrates human cognitive process models and general interaction design principles, and uses a model-based approach for systematic exploration of potential hazards. Preliminary experiments suggest that this hazard analysis technique can substantially improve the identification of use-related hazards at the early stages of software design as compared to standard hazard analysis techniques.

Categories and Subject Descriptors

D.2 [**SOFTWARE ENGINEERING**]: Requirements/Specifications; K.6.3 [**Software Management**]: Software development

General Terms

Medical systems, Hazard analysis, Use error

1. INTRODUCTION

User interface design flaws are a primary cause of use errors with medical devices [2, 4]. While user interface issues are traditionally addressed as human factors issues, there is growing evidence that they are in fact a manifestation of poor user interface software design and engineering. For example, a study carried out by engineers at the US Food and Drug Administration (FDA) highlighted that software not designed to support clinical workflows is one of the main causes of software-related recalls of medical devices [13]. In our work on the analysis of user interface software we have also identified plausible causal relationships between software design defects in medical devices and use errors [9, 6, 8, 7, 1].

To develop safe and effective medical device software, device manufacturers follow the risk management process described in ISO14971. The process includes five main steps. First, a hazard analysis is performed to identify all known and foreseeable hazards and their causes, where a *hazard* is defined as a potential source of physical injury or damage to people or environment. The standard does not mandate a specific hazard analysis technique. Second, risk estimation is performed to assess the probability of occurrence and severity of harm of each hazard, the combination of which is defined as *risk*. Third, risk evaluation is conducted to decide if every identified risk is acceptable based on pre-defined acceptability criteria. Fourth, if a risk is decided as unacceptable, control measures are designed and implemented to eliminate it or to mitigate it to an acceptable level. Finally, verification and validation activities are conducted to ensure that the designed control measures are effective. These five activities iterate and interleave until the device's overall residual risk after mitigation is acceptable.

To check that a device submitted for premarket review meets basic levels of safety, regulators need to assess the quality of the hazard analysis, as it constitutes the first step of the risk management process, and provides the basis for subsequent activities. Currently, hazard analysis techniques are not standardised in industry. In fact, each device manufacturer crafts its own version of a hazard analysis technique [3]. This creates a regulatory challenge, because hazard analysis results are hard to assess and to replicate during the premarket review process. A standard method is needed that in particular gives a strong process for identifying use-related hazards.

Contribution. We outline our ongoing work in collaboration with the FDA on establishing a reference hazard analysis technique for early identification of the causes in user interface software of use-related hazards. The aim of this work is to obtain a systematic method that can be used by manufacturers early in the development process to check that common software sources of use-related hazards have been considered and weeded out from their products. Similarly, regulators can use the same method during the pre-market review process, to check that common causes of use-related hazards have been considered by the manufacturer.

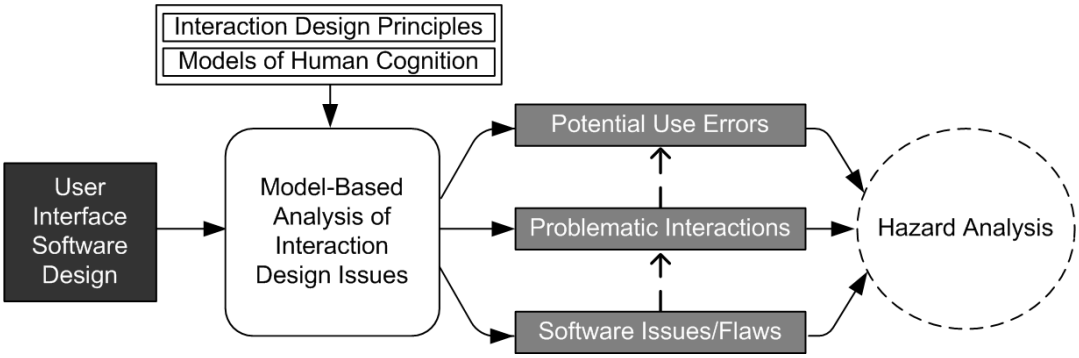

Figure 1: Overview of the analysis approach.

2. BACKGROUND

Our approach for early identification of software causes of use-related hazards combines analytical methods for hazards identification with heuristic evaluation.

Analytical methods for hazards identification are based on the idea of developing a description of the system that can be checked against tasks carried out by end users in typical use scenarios. THEA [12] is an example analytical method. The technique defines templates that provide guidance to developers for describing use scenarios and tasks. Potential issues in the system design are identified using a standard task analysis. This analysis is guided by a structured questionnaire that challenges the user interface design against general human factors concerns, such as mode visibility, and feedback. In our work, we draw ideas from THEA, and create a new analytical technique tailored for the identification of use-related hazards in software design (see Section refsec:method).

Heuristic evaluation is an analysis method used by human factors experts to evaluate user interface design against recognised design principles. Example design principles that are usually checked include: *visibility*, which is the ability of a system to make relevant information perceptible; *saliency*, defined as the ability of a system to make relevant information prominent and easy to notice; *feedback*, which is the ability of a system to provide sufficient information about what has been achieved; and *consistency*, which is the ability of a system to support similar tasks using similar operations and similar user interface elements. In our work, we use heuristic evaluation to slice the analysis of the system into focused sub-analyses that can help analysts to explore and articulate hypotheses about causal relationships between hazards and software design flaws.

3. OUR ANALYSIS METHOD

A diagrammatic view of the main steps of our analysis method is in Figure 1. The analysis takes as input user interface software design documents. A model-based analysis is thus performed for identifying use-related hazards and potential causes in software design. The output of the model-based analysis are potential use errors and problematic interactions, as well as possible causal relationships between hazards and software design flaws. These outputs are used as inputs in standard hazard analysis techniques, with the aim to extend the depth of the analysis of use-related hazards.

The model-based analysis includes three main steps:

- **Step 1: Slicing.** The aim of this first step is to slice the analysis into focused sub-analyses that facilitate reasoning about common types of use-related hazards that are commonly caused by software-related design flaws. Interaction design principles commonly used in heuristic evaluation are used to slice the analysis and ensure that at least a core set of common classes of critical problems are considered in the analysis.
- **Step 2: Identification of use hazards.** For each sub-analysis, hypotheses are formulated about use errors using models of human cognitive process and guidewords. In particular, we use the execution-evaluation model [10], and guidewords such as forgotten, wrong type, wrong order. An example use hazard generated with this method for infusion pumps is, for example, *the user forgets to configure infusion parameters*, where *forgets* is the guideword, and *configure infusion parameters* is the goal (according to the execution-evaluation model of the user's cognitive process).
- **Step 3: Identification of design issues/flaws.** For each sub-analysis, hypotheses are formulated about software design issues that could lead to use-related hazards. This is done by exploring which software features could hamper the cognitive steps followed by users (according to the execution-evaluation model) with respect to the specific design principle considered in the sub-analysis. For example, for infusion pumps, the following software design issue can be articulated for the sub-analysis related to the visibility principle: *input fields for configuring infusion parameters are rendered off screen or hidden behind a notification window when infusion parameters need to the configured.*

It is worth noticing that the basic idea behind the approach is to check that software design features create constraints that can facilitate correct trajectories of device use. This perspective is advocated in modern hazard analysis techniques such as STPA [5], as it helps to reason about logic errors in the system design — that is, cases where the system fails but its components are not faulty.

4. PRELIMINARY RESULTS

In our previous work [9], we carried out a preliminary hazard analysis focusing on the number entry software of infusion pumps. A substantial set of root causes of use-related hazards in software design was identified (results are reported in [11]). We repeated this preliminary hazard analysis using our model-based technique. We found that the technique allowed us to articulate and explore subtle causes of use hazards in a systematic way, helping us to identify additional hazards and related causes that were accidentally omitted in the original analysis. Initial experiments suggest that 3 times more use-related hazards and related causes in software design can be identified when using the model-based approach, as compared to the standard hazard analysis technique.

5. CONCLUSION

In this paper we outlined our ongoing work with the FDA on defining a systematic hazard analysis technique for early identification of potential causes in software design of use-related hazards. Our current focus is on infusion pumps and data entry software, but the technique is based on general concepts and can be applied to other devices and other user interface software components.

Acknowledgements. This work was funded by EPSRC on research agreement, EP/G059063/1. We would like to thank Paul Jones and Yi Zhang for supporting this work and contributing to its improvement.

6. REFERENCES

[1] A. Cauchi, A. Gimblett, H. W. Thimbleby, P. Curzon, and P. Masci. Safer "5-key" number entry user interfaces using differential formal analysis. In *BCS HCI*, pages 29–38. British Computer Society, 2012.

[2] A. for the Advancement of Medical Instrumentation et al. *Infusing patients safely: priority issues from the AAMI/FDA infusion device summit*. AAMI, Association for the Advancement of Medical Instrumentation, 2010.

[3] P. Jones, J. J. III, A. T. Jr., and M. Weber. Risk management in the design of medical device software systems. *Journal of Biomedical Instrumentation and Technology*, 36(4):237–266, 2002.

[4] L. L. Leape and D. M. Berwick. Five years after to err is human: what have we learned? *Jama*, 293(19):2384–2390, 2005.

[5] N. Leveson. *Engineering a safer world: Systems thinking applied to safety*. Mit Press, 2011.

[6] P. Masci, P. Oladimeji, P. Curzon, and H. Thimbleby. Tool demo: Using PVSio-web to demonstrate software issues in medical user interfaces. In *4th International Symposium on Foundations of Healthcare Information Engineering and Systems (FHIES2014)*, 2014.

[7] P. Masci, R. Ruksenas, P. Oladimeji, A. Cauchi, A. Gimblett, Y. Li, P. Curzon, and H. Thimbleby. The benefits of formalising design guidelines: a case study on the predictability of drug infusion pumps. *Innovations in Systems and Software Engineering*, 11(2):73–93, 2015.

[8] P. Masci, Y. Zhang, P. Jones, P. Curzon, and H. Thimbleby. Formal verification of medical device user interfaces using pvs. In S. Gnesi and A. Rensink, editors, *Fundamental Approaches to Software Engineering*, volume 8411 of *Lecture Notes in Computer Science*, pages 200–214. Springer Berlin Heidelberg, 2014.

[9] P. Masci, Y. Zhang, P. Jones, H. Thimbleby, and P. Curzon. A generic user interface architecture for analyzing use hazards in infusion pump software. In *Proceedings of Medical Cyber Physical Systems Workshop (MedCPS2014)*. OpenAccess Series in Informatics (OASIcs, Dagstuhl series), 2014.

[10] D. A. Norman. *The design of everyday things*. Basic books, 2002.

[11] P. Masci et al. A preliminary hazard analysis for the GIP number entry software. http://www.eecs.qmul.ac.uk/ masci/works/GIP-UI-PHA.pdf, 2014.

[12] S. Pocock, M. Harrison, P. Wright, and P. Johnson. THEA: a technique for human error assessment early in design. In *INTERACT*, volume 1, pages 247–254, 2001.

[13] L. K. Simone. Software-related recalls: An analysis of records. *Biomedical Instrumentation & Technology*, 47(6):514–522, 2013.