Human-Computer Interaction and the Formal Certification and Assurance of Medical Devices: The CHI+MED Project

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

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

Patrick Oladimeji
Swansea University
United Kingdom
p.oladimeji@swansea.ac.uk

Rimvydas Rukšėnas
Queen Mary
University of London
United Kingdom
r.ruksenas@qmul.ac.uk

Harold Thimbleby
Swansea University
United Kingdom
h.thimbleby@swansea.ac.uk

Enrico D'Urso
University of Pisa
Italy

ABSTRACT

The number of recalls of medical device with embedded computers due to safety issues in recent years suggests there is a need for new approaches to support the process. There is increasing concern about the impact of systematic use errors. There has been little research focusing on model-based tool support for the assurance and certification of medical devices with respect to systematic use error, however. The CHI+MED project (`http://www.chi-med.ac.uk`) aims to address this gap. It is concerned with the design of safer medical devices with a specific focus on human-computer interaction. We are developing a range of integrated model-based engineering methods and other formal and semi-formal techniques to support the certification process, both pre- and post-market, including their use in the wider system context. In this position paper we review our approach and the contributions to date.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification

Keywords

Model-Based System Development, Human-computer interaction

1. INTRODUCTION

As in other safety critical industries, computer-based medical equipment is subject to a process of certification before it can be marketed. Typically approaches combine pre-market review and post-market surveillance. Manufacturers are required to "demonstrate the safety and effectiveness of a device" to gain approval, with the form of the approval process depending on the degree of risk for the given class of device. Post-market surveillance involves monitoring mechanisms that identify potential safety issues with marketed devices including those that only become apparent after the device has been in service with a large user population for several years.

Most marketed medical devices are classified as medium risk 'Class II' devices [29]. They require 'special controls' that include verification against safety requirements. Medical devices are recognised as medium risk when manufacturers can demonstrate substantial equivalence to already legally marketed devices. A new device is substantially equivalent to an existing device when it has the same intended use and technological characteristics that raise no new questions of safety or effectiveness. The manufacturer can satisfy the regulator by demonstrating that the new device is at least as safe and effective as the existing device [30]. This review process is known as 510(k) clearance. Currently more than five thousand new devices require 510(k) clearance each year. The applications must be substantively reviewed within 90 calendar days of the date at which they were filed [32]. Device approval in this system is entirely based on written documents and does not involve any direct evaluation of the product. The typical size of a 510(k) submission is tens of thousands of printed pages.

The FDA is currently reviewing the 510(k) clearance process because of a series of unexpected incidents involving cleared medical devices. There are a growing number of device recalls issued for products that could cause serious health consequences including death [9]. A review of incidents with external drug infusion pumps, suggests "software errors" and "human factors errors" [31] are major concerns. This suggests that new supporting tools and processes are needed, including ones that focus on human factors errors.

There has been use of formal model-based tools in industrial contexts across a wide-range of industries dating back several decades [4]. There have been industrial strength applications of formal methods in aerospace, chip design, railways, air traffic control, nuclear industries as well as more recently by Microsoft. A wide variety of tools and techniques have been successfully used.

There is also a long history of research considering the application of formal methods to human computer interaction issues. More recently there has been research applying formal methods techniques, in some cases with a user-centred focus, specifically to medical device case studies. For example, work at the University of Pennsylvania in collaboration with the FDA is based around the development of a Generic Infusion Pump. Their aim is to develop a set of generic safety requirements for the software executed by programmable drug infusion pumps. Kim *et al.* [13], for example, applied a model-based engineering approach for generating software for a prototype infusion pump from verified specifications. Campos and Harrison [6] have used Modal Action Logic and the IVY tool to analyse the interactive behaviour of the BBraun and Alaris pumps. Bolton and Bass [3] verified a model of the Baxter iPump using SAL, including whether basic normative tasks such as sequences of actions described in user manuals are properly supported by the device. Bowen and Reeves [5], inspired by our own work, have also explored the use of ProZ for model checking safety properties of medical devices. This work suggests that model-based methods may be able to contribute to the problem of assurance and certification of medical devices.

Our project, CHI+MED, is a UK EPSRC funded project concerned with the interdisciplinary study of a wide range of aspects related to the design of safer medical devices. It brings together researchers from Queen Mary University of London (QMUL), Swansea University, UCL, and City University. Work at QMUL and Swansea has included a particular focus of developing an integrated formal model-based tool-set that supports the assurance and certification of critical medical devices. We have used infusion pumps as an exemplar device through much of the work. We are also looking at supporting the certification life-cycle both pre- and post- market. This work has been in part in collaboration with the US Food and Drug Administration (FDA) and the University of Pennsylvania.

We have been working on a range of separated strands with the long term aim of drawing them together into an integrated and accessible tool set that supports the development of verifiably safer medical devices.

Generic Reference Architecture In section 2 we describe our development of a reference architecture for infusion pumps. This idea is central to several of the other approaches.

Formal Safety Requirements In section 3 we describe our work on formalising safety requirements including a hazard analysis based around the reference architecture. Such formal safety requirements are the basis for the remainder of our work. We are building on a preliminary set of requirements developed by the FDA.

Model-based Engineering of user interfaces The core of our work is exploring model-based engineering approaches to the verification of medical devices from formal requirements. This work is overviewed in section 4.

Developing user interface prototypes We are also investigating ways to make the tools developed accessible to actual existing development processes and in particular the rapid prototyping of user interfaces from formal models. This work (described in section 5) has also involved the integration of control and user interface models developed in different formalisms.

Modeling the wider system Whilst the above strand focuses primarily on the devices themselves, a further strand (section 6) is looking at how modelling of the wider system can support not just verification, but also understanding of devices in use, that could be of benefit in post-market review, ultimately leading to new safety requirements.

2. REFERENCE ARCHITECTURE

Much of our work linked to verification and certification is based around the development of generic reference specifications for the medical devices under consideration. This approach follows that adopted by the University of Pennsylvania with the FDA on the Generic Infusion Pump project. It aims to develop safety models that can be used as a reference specification verified with respect to formal safety requirements [1]. Such a reference model can, for example, demonstrate how exemplar designs can in practice meet a given set of requirements. It can also be used to generate test cases, and be the basis for model-based engineering approaches to development.

The University of Pennsylvania work has focused primarily on the (internal) control system of the generic infusion pump and only included an abstract human-computer interface. Our work has developed this part of the reference model, with a focus on the part of the wider system relevant to safety requirements intended to prevent or otherwise manage use hazards, such as systematic human error in the use of the devices [24].

To date we have developed a Generic Infusion Pump User Interface (GIP-UI) architecture the software components of which are outlined in Figure 1. It is organised around documentation, physical widgets (e.g., the actual buttons and display), their software drivers, and a set of core software modules. The core modules consist of interaction logic that processes input events and determines the feedback, the output status manager that determines what feedback is presented to the user, and the renderer that determines how information is presented. Finally, a further input interpreter which deals with more complex input mechanisms that, unlike physical buttons, need processing to determine what is being input such as gestures on a touch screen. The diagram focuses only on software components, though the full reference architecture includes other hardware pathways. For example, user interfaces may include safe emergency buttons that are purely electrical/hardware controls (e.g., an emergency stop button). These buttons directly change the hardware state of the device. In these situations, the GIP-UI software will receive notifications about new device states from the GIP controller, which listens for hardware events, and updates the user interface state accordingly.

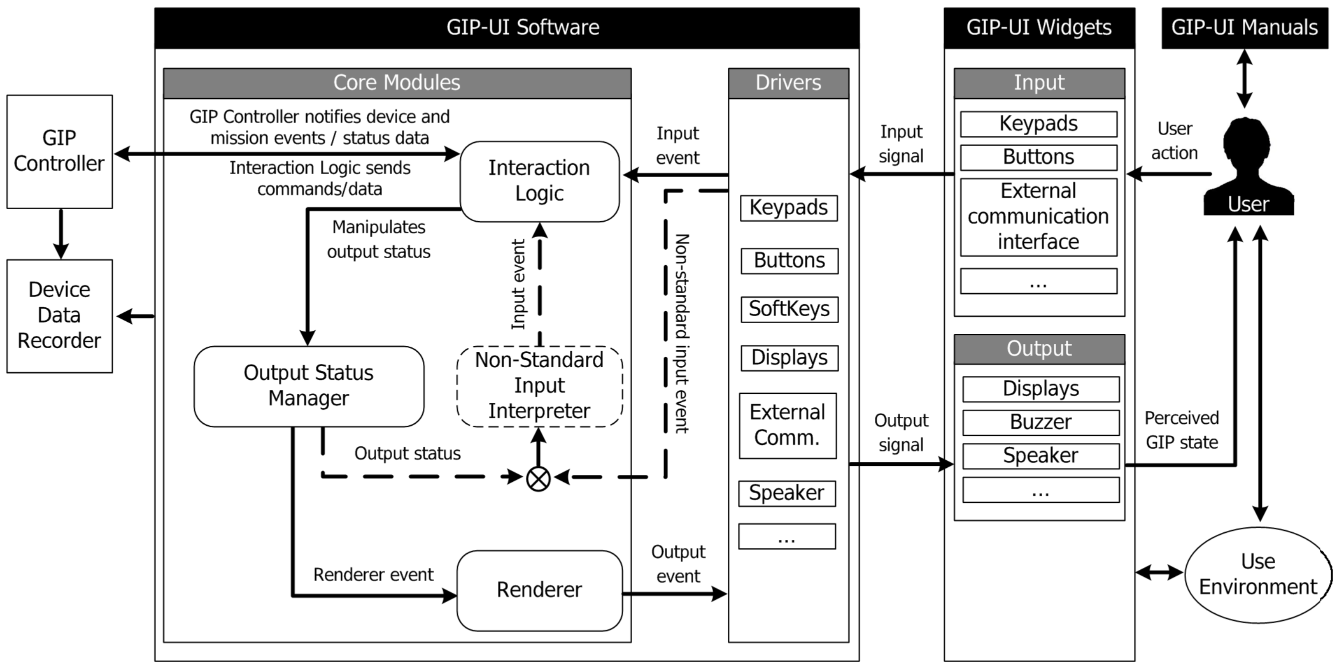

Figure 1: Architecture of the GIP user interface (GIP-UI) showing key functional components of the system.

The architecture can be used to reason about design defects in infusion pump user interface software that may potentially cause use hazards [24]. We have used it to support a preliminary hazard analysis, as described in section 3, which demonstrates its utility. It has also been used as part of a model-based engineering process. In particular we have used it to create an initial prototype, verified against a selected set of safety requirements, discussed in section 4.

3. FORMAL SAFETY REQUIREMENTS

The techniques we are developing are founded on having formally specified safety requirements concerned with human-computer interaction issues. Such a set would differ for each device and depend on the context in which the device was to be used. However there are likely to be many commonalities: for example safe number entry is likely to lead to a core set that are common. Currently a comprehensive set of such requirements for devices such as infusion pumps do not exist even informally. The FDA have developed an initial set of safety requirements for infusion pumps based on software hazards. However, it only partially addresses *use* hazards. A more complete set of safety requirements are needed.

The first step in the development of such formal safety requirements, then, is to conduct a hazard analysis around which a complete set of requirements can be developed [12]. Once a set of safety requirements is obtained they then need to be formalised, and the formal versions verified against the natural language versions.

3.1 A Preliminary Hazard Analysis

As this first step in the development of a hazard analysis of use hazards we have conducted an analysis of infusion pump incident reports and other information sources including earlier CHI+MED research (e.g., [20]). This has led to a review of the common use hazards in infusion pumps on the market related to number entry tasks, and the completion of a preliminary hazard analysis [14, 24].

We used the formal user interface reference architecture, described in section 2, to identify and reason about design errors commonly present in infusion pump number entry software and their relation to use hazards. In particular, this allowed us to better understand the ramifications of the use hazards identified and to make concrete links to reported software design errors that could cause them. While this analysis focused on the number entry sub-systems of infusion pumps, much of the analysis is relevant to similar subsystems of other devices. Examples include ignored key presses (e.g., decimal point key presses being ignored in some situations leading to out by 10 errors in drug doses) and inappropriate and incorrect feedback.

Having identified a large number of use errors and potential flaws, we conducted an informal survey of other devices. This involved examining a range of devices in a major hospital, together with other devices we had access to. The main focus of the hazard analysis was on number entry, so we targeted devices with number entry interfaces. In addition to a general, if limited exploration of the interface, we inputed sequences known to be problematic from the hazard analysis and our other work (e.g., entering double decimal points and checking boundary cases). This showed that similar design flaws crop up in many devices, from many different manufacturers and across many different classes of device, not just infusion pumps. Over 30 software design issues that could have serious clinical consequences were discovered in the hazard analysis. The issues were observed across 9 medical devices from 6 different manufacturers. This suggests the problem is systemic and certainly not just a problem for any single manufacturer.

As a consequence of finding such a large range of problems we were asked by the hospital to develop a training video for clinicians that overviews specific problems to look for [15].

3.2 Formalising safety requirements

Our work developing a comprehensive set of use error based safety requirements for infusion pumps is still in progress. In parallel, we have explored approaches to the formalisation and use of such human factors based safety requirements based on a small set of existing requirements. In particular, an FDA draft document [2] includes safety requirements for PCA pumps, some of which are related to use error. They were intended to be easily translated into a specification in either a programming or formal language.

Examples include:

- "Clearing of the pump settings and resetting of the pump shall require confirmation."
- "The pump shall issue an alert if paused for more than t minutes."

We have developed a process for formalising safety requirements based around a generic reference model [16]. We first identify terms in the safety requirements, such as 'clear settings', 'reset pump', and 'require confirmation', that specify functionalities of the reference architecture. These are represented as uninterpreted predicates on the machine state at this stage. They are combined into logical statements, corresponding to the intended semantics of the requirement. These statements are still abstract as they do not include a semantics for the terms introduced. They are uninterpreted only in the logical sense - their link to the requirements gives them an informal interpretation that should be made explicit. The logical statements that contain them give formal relational constraints that must be verified to meet the corresponding safety requirement. These constraints are then formed into PVS axioms specifying a state transition system in a form that supports structural induction: that the constraint holds of the initial state and is preserved by the transitions. This gives an abstract reference model of any device intended to meet the requirements. It gives a formal semantics to the generic architecture based directly on the requirements. Such an abstract reference model is also a suitable format for performing model refinement as discussed in section 4.

At this point the formalised safety requirements are not device specific. A manufacturer would need to instantiate them to a form consistent with the user interface of a particular device to be certified. Part of the assurance case would then be an argument that the instantiation chosen is appropriate.

4. MODEL-BASED ENGINEERING

4.1 Modelling a device user interfaces

Given a formalised set of requirements the next step in a model-based development process is to create a formal model specifying the user interface of the device. In our process this model specifying a concrete device is given as a state transition machine. Actions on the interface are specified as transition functions over states of the user interface. We have modelled several commercial infusion pumps at various levels of abstraction. We have also developed a preliminary user interface prototype for a Patient Controlled Analgesia (GPCA) pump based on the reference architecture described in section 2 in this way.

Developing PVS specifications can be a complex, time consuming and error prone process, however. To address this, we have developed a design support tool, Emulink [8]. It allows users to develop formal PVS specifications using a graphical formalism similar to Statecharts. Emulink presents itself as a typical Statecharts editor environment with functionalities such as creating nodes and adding transitions. As soon as users add graphical objects on the diagram a PVS specification, in a form suitable to fit the rest of our process, is created and updated, reflecting the behaviour of the diagram. It is configured as a plug-in for our development environment PVSio-web described in section 5 below. This allows the underlying PVS specification created by Emulink to be animated. During device simulation, the current and previous Emucharts states are highlighted so that users can follow the device behaviour both looking at the picture and looking at the Emulink diagram. The system currently supports the following graphical objects: States, Transitions, Self Transitions, Default Transitions, Transition Condition, Transition Action. This has been sufficient for our purposes of modelling trialling the system. However in future work we will extend the subset with features such as parallel states and connective junctions. Emulink is similar in concept to, for example, the DOVE tool [7] for the Isabelle proof system. Our main contribution is in the way that we are integrating it with PVS-based tools for the model-based engineering of interfaces.

4.2 Verifying behavioural specifications

At this point in the process we have a concrete specification of a particular device's user interface. We also have a set of safety requirements in the form of an abstract model with the key terms uninterpreted. The next step is to give semantics to the uninterpreted terms based on the device user interface model. In our process this is done using PVS *theory interpretation* [17, 16]. This involves importing the safety requirements theory into that of the device model. As part of the importing process we give definitions for each of the uninterpreted terms. In doing so PVS automatically generates obligations to prove that each newly created concrete version of the axiom is true. Discharging these proof obligations demonstrates that the concrete model meets the given safety requirements.

We have trialled this general approach in several ways. We have used this process to formally verify our GPCA pump prototype specification against the formalised set of FDA user based safety requirements demonstrating the feasibility of the approach at least to verify that a basic level of safety is met [17]. We have also applied the verification approach to a commercial patient controlled analgesia pump [16]. In this case for practical reasons the specification of the interface was reverse engineered from a combination of exploring the behaviour of actual pump and the user manual. Whilst such reverse engineering would not be done by a manufacturer, this case study does demonstrate the verification approach is applicable to a real device.

4.3 Refinement from requirements

Stepwise refinement techniques provide an alternative way to demonstrate that a formalised set of requirements is satisfied by the user interface of a device that we are exploring. We are using the Event-B/Rodin platform to develop usage models [11] that are correct by construction. A usage model is a formal representation that describes the common characteristics and behaviour of software for broad classes of devices. The goal is to develop usage models that satisfy core sets of safety requirements that can mitigate against typical hazards. Event-B is used to express the high level safety requirements such as those proposed by the FDA. Refinement is used to demonstrate that the requirement can be cascaded into a hierarchy that encompasses different input/output technologies. We have proposed a refinement approach based on three verification layers [27].

In the requirements layer, a minimal set of requirements, relevant to some safety aspects of device interfaces, is first developed. The aim is that these requirements should be sufficiently abstract to encapsulate the behaviour of the largest class of possible devices. Refinements are then used to detail these requirements in a sequence of steps. It is also possible that refinement can lead to alternative interface requirements that also provide assurance of the safety of the device. These modified requirements would be developed as a contract between regulator and manufacturer. Having produced an abstract version of the requirements, the next stage is to make sense of them in terms of the particular devices that the developer wishes to certify.

In the interface hierarchy layer, a refinement-based classification of user interfaces has been developed that is relevant for various modes of data entry in infusion pumps. Each refinement step introduces specific features, thereby creating a hierarchy of user interface classes. The aim is that requirements are verified once for the most abstract classes of interfaces. More concrete classes of interfaces at the lower levels of this hierarchy are then guaranteed to satisfy the requirements by construction.

Having an interface hierarchy already verified against the relevant requirements simplifies the process of demonstrating that a specific interface satisfies these requirements. In the concrete interface layer, it suffices simply to demonstrate that the specific interface is an instance of some class in the interface hierarchy.

4.4 Verifying existing implementations

The ideal is that model development (and verification) is directly integrated into the design process. However, that does not match the reality of the way code for medical devices is currently developed. Even in the longer term there is liable to be legacy code that needs to be used and so assured. We have also therefore explored ways of using formal methods for the assurance of existing user interface software. In this situation a manufacturer (or regulator) would not reverse engineer the interface but have the source code.

To this end, in collaboration with the FDA, we developed an approach that allows design issues related to user interaction to be detected in existing software [22, 21] based around configuration diagrams [28].

The first step is to translate the source-code implementation of user interface software into an equivalent formal specification. This is done by a direct translation of the source code constructs into PVS. From this a behavioural model, in the form of a configuration diagram, is constructed using theorem proving. Configuration diagrams are a form of transition diagram based specification, built directly around the invariant property to be verified. Nodes, or configurations, correspond to groups of states for which both the invariant and some other property specific to that node hold. Edges are conditions that transition the state between the configurations. In our use the invariants are interaction design properties such as consistency of actions and feedback. They would ultimately be derived from safety requirements. The configuration diagram is created node by node by symbolically executing the implementation model from an initially specified configuration that is reachable from the initial state of the device. As each new node in the diagram is created, it is verified whether the property holds of it or not.

Sequences of test inputs can then be produced for situations where the invariant does not hold, by exploration of the configuration diagram. These test cases can be executed on the actual implementation to ensure that the issues detected in the behavioural model do apply to that implementation. This addresses the problem that the model cannot be guaranteed to be an accurate description of the implementation, at least with respect to false positives.

We have trialled the approach by using PVS to analyse the user interface of an existing commercial medical device implemented in C++. In doing so we found several previously unknown interaction design issues in the device, which may potentially lead to severe consequences. Problems discovered include: valid input key sequences that are incorrectly registered without the user's awareness; inappropriate feedback being given to the user for error conditions; ill-formed input key sequences that are silently accepted without the user's awareness; and digits keyed after a decimal point that are silently discarded without making the user aware [22, 21].

Ultimately such verification should not be against ad-hoc properties, but against properties derived from formal safety requirements as set by the regulator.

5. EXECUTABLE PROTOTYPES

In practice, multiple stakeholders are involved in the development, procurement and regulation of a product. For formal verification to be a useful and effective tool, effective communication between formal verification experts and stakeholders both within development teams and externally, such as with regulators, is needed. That requires integrated tools that enable the use of verification techniques while hiding the complexity involved in creating and interacting with classic verification tools. One way to bridge this communication gap is to be able to quickly create prototypes from the engineered models to allow the behaviour of a specification to be explored in a way that can be easily understood. For human-computer interaction issues that means an actual interactive mock-up of the interface. Such a prototype allows early validation of the interaction design and can be used, for example, for more traditional usability evaluation. Changes to the specification can then be quickly mirrored in executable models. To date we have developed a way to rapidly prototype user interfaces driven from the formal model and to combine the execution of the prototypes with control logic developed in Simulink.

5.1 Prototyping user interfaces

We have developed PVSio-web [25], as an environment that facilitates the easy creation of realistic prototypes for interactive systems that are driven by PVS specifications. The environment allows animation of a PVS specification using a graphical user interface corresponding to the interactive device being modelled. Users define interactive input and output areas over a picture of the interactive device being modelled. Input areas drive commands that are executed using PVS and the output areas are used to visualise state information about the interactive system. This is built upon the PVSio simulation environment of PVS.

We have developed a variety of executable models in PVSio-web, including a prototype of our GPCA specification. This demonstrates the final step of a model-based engineering approach that ultimately develops an executable prototype of a user interface derived from a reference architecture and that is verified against a set of safety requirements. We have also used PVSio-web as a simulation tool to visualise problematic input sequences with safety consequences on a variety of devices, illustrating the use of the approach as a way of communicating results of verification to non-verification expert stakeholders.

The focus of our work has been on verification against requirement specifications. The integration with such prototyping and simulation tools can also be used to support validation against user needs. Such tools give the design team an early opportunity to validate the models.

5.2 Integrating PVS and Simulink

Developing new verification tools, however powerful, is not enough for them to be usefully adopted in practice. A design process might typically involve a range of different models and tools, as different tools have different strengths and weaknesses. New tools need to integrate with existing combinations of design, modeling and verification tools. For example, MathWorks Simulink is a modeling framework widely used in industry and can be used to develop control software for medical devices. As with PVS which we use to develop models of the human-computer interfaces of devices, it has native simulation tools. Irrespective of the formal verification approach used, simulation is a standard approach to help validate models. Therefore, ways to co-simulate models developed in each framework are needed.

We have developed a new, flexible approach for integrating PVSio (the simulation tool of PVS) with Stateflow, the Simulink control logic tool [23]. It establishes web services to create a communication infrastructure between the two frameworks. This allows simulations of control and interface models to be executed in parallel each driven by communication of events from the other. It also opens the possibility for the wide range of applications developed in Stateflow to benefit from the rigor of PVS verification.

We have illustrated its use with a simple case study involving the GPCA pump prototype, the interface for which we developed as described in the previous sections. We combined it with a software controller that had previously been developed in Stateflow as part of the infusion pump project. Simulation of the prototype demonstrated that the PVSio and Stateflow components inter-operate effectively. Our tools and example models are available at `http://www.pvsioweb.org`.

6. MODEL-BASED UNDERSTANDING OF THE WIDER SYSTEM

The research described so far focuses primarily on the device and its interface. There is potential benefit in considering more explicitly the wider system including the users and their cognitive limitations, other artifacts (interactive devices, user manuals), other people that relevant information flows through, as well as the wider socio-technical system. We have therefore also been exploring ways that aspects of this wider system can be modelled and reasoned about in ways that support the assurance of the device under consideration.

6.1 Modelling cognitive assumptions

The first step outwards from the device is to consider the behaviour of the user and in particular cognitive limitations that lead to systematic error. When evaluating an interactive system design from this perspective it is necessary not only to formally describe assumptions about the device design but also the assumptions that are being made about the user in terms of their cognitive capabilities and context.

We have developed a modelling framework [26] that can be used to highlight error prone interaction design given such a set of cognitive assumptions. We have also provided an approach to exploring the consequences of such assumptions that combines formal verification techniques with laboratory-based empirical studies [26].

The main idea here is to model, and explore the consequences of, assumptions about how medical devices will be used [26]. Abstract models of prototype designs and use assumptions can then be analysed using model checking techniques in order to predict, before deployment, design aspects that might compromise the safety and usability of a device.

The purpose of the user model is to restrict the device behaviours to those that are consistent with user behaviour given the cognitive assumptions. In our approach, the specific user model analysed is based on an instantiation of a generic user model. This generic model provides the means to use different sets of cognitive assumptions to allow different scenarios to be considered. It can be instantiated with the particular task assumptions relevant to the analysis. Used in conjunction with empirical study of device use, this makes it possible for the experimenter to make, and explore, conjectures about how the device will be used. We believe that this approach gives increased analytical power to empirical results.

We have also used a formal model of cognitive assumptions to undertake a more fine-grained analysis based on information needs and the resources afforded by a designed system.

6.2 Understanding actual use in context

The certification process does not in reality end at the point a device is brought to market. Ultimately, they are used in, and their use adapted to, specific contexts whether in a hospital or home. This can lead to them being used in ways or contexts that were not imagined when certified. Also issues arise due to novel use situations or just unforeseen circumstances that can lead to accidents and so potentially recalls. Approaches are then needed for understanding device use in such wider socio-technical contexts, either to determine ways to improve the system after incidents, or in a more preemptive way to understand practice as it develops or based on a precursor device.

To do this in a formal way involves modelling more than just individual interfaces and users but, for example, modelling artifact rich contexts and information flow through them. We have done some preliminary exploration of the development of formal models and proof-based tools that support taking a wider socio-technical view of medical device design [19, 18].

One strand of work has been to model information resources and their transformation through the wider socio-technical system within which a device is used. A variety of views can be taken — what documentation says happens, what actually happens determined by observation, what stakeholders claim happens, etc. By modelling the different processes as described by different sources and comparing them we can highlight potential areas of concern.

Specifically, we developed a constructive method to support incident investigation [19]. It centres on models of information resources used by those involved in the incident (such as the infusion rate printed on a medication order), together with models of how information resources are transformed as they propagate through the system (e.g., how a medication order is entered into the pharmacy information system). Once the information resources have been developed we formulate and verify conjectures about how resources were used (such as that relevant resources were available at critical moments to relevant actors) and facts about the prescribed use of information resources based on procedures and regulations. We have applied this method, using PVS, to the details of an actual incident where an infusion pump was incorrectly programmed leading to it delivering an incorrect drug dose to a patient in an oncology out-patients unit [10]. The analysis highlighted potential areas for improvement of the system not mentioned in the report. For example, neither the label nor the pump provided information about safe limits: it was not easily available to the nurse at the point when it was needed. This shows that such an approach can potentially help detect safeguards and weaknesses.

We have also explored how a relatively simple use of PVS can support a field researcher understanding a medical system. In particular it can help externalise assumptions and facts, verify the consistency of the logical argument framed in the descriptions, help uncover latent situations that may warrant further investigation by the field researcher, and verify conjectures about potential hazards linked to the observed use of information resources. This approach was originally applied to a medical dispatch system [19], but has more recently been used to support a field researcher studying the introduction of a new glucometer within a hospital ward context.

7. CONCLUSIONS

A key long term goal of the CHI+MED project is to make formal techniques accessible to developers so they can be practically used within the certification/assurance process. The intention is that they both focus on the device and the wider systems and processes in which they are embedded. This involves developing an integrated and accessible toolkit that combines theorem proving, model checking, simulation and visualisation techniques based around formal HCI safety requirements.

We have shown how user interfaces can be developed using a model-based engineering approach. Starting with a reference architecture for the device in question, a hazard analysis leads to the development and formalisation of safety requirements. Abstract versions of these requirements would ultimately be issued by the regulator which would require manufacturers to demonstrate they met them. The manufacturer would develop concrete versions of the requirements for particular devices in negotiation with the regulator.

We also present approaches a manufacturer could use to then verify both new and legacy code against the requirements. The model specifying the user interface behaviour that is verified can also form the basis of an interactive prototype of the interface that inter-operates with Stateflow models of the device's control software. We have created tools that support this process. We have also shown its feasibility by developing a prototype of a PCA pump, verified against safety requirements, as well as applying elements to commercial infusion pumps.

We have also shown how model-based approaches can support empirical work aimed at understanding the wider system in which medical devices operate, including exploring the ramifications about limitations of users and the way information propagates around the system.

8. FURTHER WORK

The approaches described have mainly been applied to the case studies around which they were developed. In future we need to apply the toolkit to a wider range of medical case studies around different classes of device, and further explore its integration into the certification/assurance process. A key prerequisite of this is to develop and formalise a comprehensive set of user-based safety requirements for the devices concerned. To date we have focused mainly on number entry systems and infusion pumps. We need to explore wider aspects of infusion pumps and wider classes of devices.

An area needing attention is for devices where complex and potentially remote data entry is involved. The research carried out so far has ignored features of new technologies where the devices are driven by data — for example Drug Error Reduction (DER) software uses pharmaceutical databases to reduce the likelihood of mistaken data entry at the bedside. Other technologies, for example radiotherapy machines, are driven by complex prescribed patient data. Another future direction is to apply similar approaches to the security and integrity of the data and the devices.

Our existing tools make it possible to visualise the scenarios and to prove that certain properties of the wider system are true, subject to information resource constraints. We need to explore the possibilities here in more depth. We also propose to go a step further building on existing work to produce stochastic simulations or fluid flow models that enable consideration of typical uses of the system. This will enable a consideration of issues associated with the performance of the system.

The resulting multi-level toolkit could be used to support system design, system redesign linked to incident investigation and pre-emptive system redesign. Different methods, tools and levels of formality are needed for different aspects of the socio-technical system and exactly what works at the different levels will be a focus of future research.

An additional area to look at is the integration of our tools and approaches with existing standards such as IEC 62366: 2007. It specifies a process for manufacturers to analyse, specify, design, verify and validate usability with respect to safety of a medical device. However, it follows a very traditional usability engineering approach and does not provide ways to look at exhaustive coverage as our model-based approaches potentially do. Model-based approaches such as ours could help plug this major gap.

An additional interesting direction would be to apply our techniques to analyse the consequences of possible actions of users in situations where hazards have arisen including in the event of system errors such as software bugs manifesting.

9. ACKNOWLEDGMENTS

This work is supported by EPSRC as part of CHI+MED (Computer-Human Interaction for Medical Devices, EPSRC research grant [EP/G059063/1]). We are grateful to Michael Harrison, Yi Zhang and Paul Jones, Anaheed Ayoub, Insup Lee, Oleg Sokolsky as well as members of CHI+MED across the four sites, who collaborated in different aspects of the work described. We thank the anonymous referees who made a series of very helpful comments and suggestions that improved the paper.

10. REFERENCES

[1] D. Arney, R. Jetley, P. Jones, I. Lee, and O. Sokolsky. Formal methods based development of a PCA infusion pump reference model: Generic infusion pump (GIP) project. In *Proceedings of the High Confidence Medical Device Software and Systems (HCMDSS)*, pages 23–33, June 2007.

[2] D. Arney, R. Jetley, P. Jones, I. Lee, and O. Sokolsky. Generic Infusion Pump Hazard Analysis and Safety Requirements, February 2009. Technical Report MS-CIS-08-31.

[3] M. Bolton and E. Bass. Formally verifying human—automation interaction as part of a system model: limitations and tradeoffs. *Innovations in Systems and Software Engineering*, 6(3):219–231, Sept. 2010.

[4] J. Bowen and M. Hinchey. The use of industrial-strength formal methods. In *Proceedings of the 21st Computer Software and Applications Conference, (COMPSAC '97)*, pages 332 – 337, 1997.

[5] J. Bowen and S. Reeves. Modelling safety properties of interactive medical systems. In P. Forbrig, P. Dewan, M. Harrison, and K. Luyten, editors, *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS2013)*, pages 91–100, 2013.

[6] J. Campos and M. D. Harrison. Modelling and analysing the interactive behaviour of an infusion pump. *Electronic Communications of the EASST*, 45: Formal Methods for Interactive Systems 2011 (FMIS 2011), June 2011.

[7] A. Cant, K. Eastaughffe, C. Liu, B. Mahony, J. McCarthy, and M. Ozols. State-machine modelling in the DOVE system. DSTO Research Report, DSTO-RR-0255, February 2003.

[8] E. D'Urso. *Emulink: a graphical modelling environment for PVS*. University of Pisa, 2014. MSc Thesis.

[9] Institute of Medicine. Medical Devices and the Public's Health: The FDA 510(k) Clearance Process at 35 Years, 2011.

[10] ISMP Canada. Fluorouracil incident root cause analysis report, 2007. `http://www.ismp-canada.org/download/reports/FluorouracilIncidentMay2007.pdf`.

[11] R. Jetley, S. Purushothaman Iyer, and P. Jones. A formal methods approach to medical device review. *Computer*, 39(4):61–67, 2006.

[12] P. Jones, J. Jorgens, A. J. Taylor, and M. Weber. Risk management in the design of medical device software systems. *Biomedical Instrumentation and Technology*, 36(4):237–266, July-August 2002.

[13] B. Kim, A. Ayoub, O. Sokolsky, I. Lee, P. Jones, Y. Zhang, and R. Jetley. Safety-assured development of the GPCA infusion pump software. In *Proceedings of the ninth ACM international conference on Embedded software*, EMSOFT '11, pages 155–164. ACM, 2011.

[14] P. Masci. A preliminary hazard analysis for the GIP number entry software, 2014. CHI+MED technical report available from `http://www.chi-med.ac.uk/researchers/bibdetail.php?docID=730`.

[15] P. Masci. Design issues in medical user interfaces , 2014. Video available from `http://www.chi-med.ac.uk/researchers/bibdetail-db.php?PPnum=IP004`.

[16] P. Masci, A. Ayoub, P. Curzon, M. D. Harrison, I. Lee, and H. Thimbleby. Verification of interactive software for medical devices: PCA infusion pumps and FDA regulation as an example. In P. Forbrig, P. Dewan, M. D. Harrison, K. Luyten, C. Santoro, and S. D. J. Barbosa, editors, *Proceedings of the ACM SIGCHI*

Symposium on Engineering Interactive Computing Systems, EICS'13, pages 81–90. ACM, June 2013.

[17] P. Masci, A. Ayoub, P. Curzon, I. Lee, O. Sokolsky, and H. Thimbleby. Model-based development of the generic PCA infusion pump user interface prototype in PVS. In *Proceedings of the 32nd International Conference on Computer Safety, Reliability and Security (SafeComp 2013)*, September 2013.

[18] P. Masci, P. Curzon, D. Furniss, and A. E. Blandford. Using PVS to support the analysis of distributed cognition systems. *Innovations in Systems and Software Engineering*, April 2013. Published online.

[19] P. Masci, H. Huang, P. Curzon, and M. D. Harrison. Using PVS to investigate incidents through the lens of distributed cognition. In *Proceedings of the 4th international conference on NASA Formal Methods*, NFM'12, pages 273–278, Berlin, Heidelberg, 2012. Springer.

[20] P. Masci, R. Rukšėnas, P. Oladimeji, A. Cauchi, A. Gimblett, Y. Li, P. Curzon, and H. W. Thimbleby. The benefits of formalising design guidelines: A case study on the predictability of drug infusion pumps. *Innovations in Systems and Software Engineering*, April 2013. Published online.

[21] P. Masci, Y. Zhang, P. Curzon, M. D. Harrison, P. Jones, and H. Thimbleby. Verification of software for medical devices in PVS. CHI+MED Technical Report, 2013.

[22] P. Masci, Y. Zhang, P. Jones, P. Curzon, and H. Thimbleby. Formal verification of medical device user interfaces using PVS. In *ETAPS/FASE2014, 17th International Conference on Fundamental Approaches to Software Engineering*, Berlin, Heidelberg, 2014. Springer-Verlag.

[23] P. Masci, Y. Zhang, P. Jones, P. Oladimeji, E. D'Urso, C. Bernardeschi, P. Curzon, and H. Thimbleby. Combining PVSio with Stateflow. In *Proceedings of the 6th NASA Formal Methods Symposium (NFM2014)*, Berlin, Heidelberg, April-May 2014. Springer-Verlag.

[24] 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 V. Turau, M. Kwiatkowska, R. Mangharam, and C. Weyer, editors, *5th Workshop on Medical Cyber-Physical Systems*, volume 36 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–14, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[25] P. Oladimeji, P. Masci, P. Curzon, and H. Thimbleby. PVSio-web: a tool for rapid prototyping device user interfaces in PVS. In J. Bowen and S. Reeves, editors, *Proceedings of the 5th International Workshop on Formal Methods for Interactive Systems*. EASST, June 2013.

[26] R. Rukšėnas, P. Curzon, and M. D. Harrison. Integrating formal predictions of interactive system behaviour with user evaluation. In E. B. Johnsen and L. Petre, editors, *Proceedings of Integrated Formal Methods: LNCS 7940*, pages 238–252. Springer-Verlag, June 2013.

[27] R. Rukšėnas, P. Masci, M. D. Harrison, and P. Curzon. Developing and verifying user interface requirements for infusion pumps: A refinement approach. In J. Bowen and S. Reeves, editors, *Proceedings of the 5th International Workshop on Formal Methods for Interactive Systems*. EASST, June 2013.

[28] J. Rushby. Verification diagrams revisited: disjunctive invariants for easy verification. In *Proceedings of Computer Aided Verification (CAV2000)*, pages 508–520. Springer, April-May 2000.

[29] US Food and Drug Administration. Learn if a Medical Device Has Been Cleared by FDA for Marketing, 2009.

[30] US Food and Drug Administration. Premarket Notification (510k), 2009.

[31] US Food and Drug Administration. Total Product Life Cycle: Infusion Pump - Premarket Notification [510(k)] Submissions - Draft Guidance, April 2010.

[32] US Food and Drug Administration. FDA and Industry Actions on Premarket Approval Applications (PMAs): Effect on FDA Review Clock and Goals, October 2012.