chi+med
making medical devices safer

EPSRC Programme Grant EP/G059063/1

Public Paper no. 211

A generic user interface architecture for analyzing use hazards in infusion pump software

Paolo Masci, Yi Zhang, Paul Jones,
Harold Thimbleby & Paul Curzon

Masci, P., Zhang, Y., Jones, P., Thimbleby, H., & Curzon, P. (2014). A generic user interface architecture for analyzing use hazards in infusion pump software. *Proceedings of the 5th Workshop on Medical Cyber-Physical Systems (MCPS 2014)*, 1–14. OASIcs: Open Access Series in Informatics, vol. 36.

PP release date: 28 March 2014

file: WP211.pdf

UCL UCLIC Swansea University Prifysgol Abertawe FIT LAB Queen Mary University of London CITY UNIVERSITY LONDON EPSRC Engineering and Physical Sciences Research Council

A Generic User Interface Architecture for Analyzing Use Hazards in Infusion Pump Software

Paolo Masci[1], Yi Zhang[2], Paul Jones[2], Harold Thimbleby[3], and Paul Curzon[1]

1 **Queen Mary University of London**
London, United Kingdom
{p.m.masci,p.curzon}@qmul.ac.uk
2 **Center for Devices and Radiological Health**
U.S. Food and Drug Administration, Silver Spring, MD, USA
{yi.zhang2,paul.jones}@fda.hhs.gov
3 **College of Science, Swansea University**
Swansea, Wales, United Kingdom
h.thimbleby@swansea.ac.uk

Abstract

This paper presents a generic infusion pump user interface (GIP-UI) architecture that intends to capture the common characteristics and functionalities of interactive software incorporated in broad classes of infusion pumps. It is designed to facilitate the identification of use hazards and their causes in infusion pump designs. This architecture constitutes our first effort at establishing a model-based risk analysis methodology that helps manufacturers identify and mitigate use hazards in their products at early stages of the development life-cycle.

The applicability of the GIP-UI architecture has been confirmed in a hazard analysis focusing on the number entry software of existing infusion pumps, in which the GIP-UI architecture is used to identify a substantial set of user interface design errors that may contribute to use hazards found in infusion pump incidents.

1998 ACM Subject Classification K.6.1 Systems Analysis and Design, H.5.2 User Interfaces, D.2.11 Software Architectures

Keywords and phrases Infusion Pump, Hazard Analysis, Use Hazards, User Interface, Interactive Software, Design Errors

Digital Object Identifier 10.4230/OASIcs.MCPS.2014.1

1 Introduction

Infusion pumps are a class of medical devices widely used in various clinical settings to deliver fluids (including medication and nutrients) into a patient's body in a controlled (prescribed) manner. The safety of infusion pumps, however, has been one of the top concerns in health care for a number of years [7]. For example, during the period from 2005 to 2009, 87 recalls associated with infusion pumps were reported in US due to defective design or manufacturing problems [8]. Through the analysis of incidents involving infusion pumps, medical device regulators such as the US Food and Drug Administration (FDA) concluded that two of the major factors contributing to infusion pump failures were *software defects* and *user interface issues* [33].

Many user interface issues can be associated with software problems. For instance, a key bounce may be caused by a defect in the interrupt-handling code, and problems

Medical Cyber Physical Systems – Medical Device Interoperability, Safety, and Security Assurance (MCPS'14).
Editors: Volker Turau, Marta Kwiatkowska, Rahul Mangharam, and Christoph Weyer; pp. 1–14

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

related to configuring the pump may be caused by inappropriate user-device interaction logic. However, there are also user interface issues that have roots in the software engineering process. For example, user interface problems may arise if the device's interaction behavior is inconsistent with its user manual or labeling. Either type of user interface issues can affect the device operation and hence affect patient safety. To help reduce or eliminate these issues, a systematic risk management process should be carried out.

ISO14971:2007 is a standard that provides a systematic risk management process for medical devices. In brief, it consists of five distinct activities. 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. 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.

In the ISO14971 risk management process, the identification of hazards constitutes the first step and provides the basis for subsequent activities. In our previous work [10, 36], we illustrated the benefits of using a Generic Infusion Pump (GIP) architecture to support a systematic hazard analysis early in the design process. The GIP architecture, which captures common characteristics and functionalities of broad classes of infusion pumps, serves as a reference 'standard' for reasoning about hazards and potential causal factors in infusion pump designs.

In this paper, we present an effort that extends our previous work focusing on the user interface design and its associated use hazards. In particular, we extend the GIP architecture with more details that reflect common user-device interaction designs in existing infusion pumps. This serves to establish a foundation for reasoning about use hazards in infusion pumps and their contributing factors rooted in defective interaction (software) design.

Contributions. The contributions of this paper are as follows: (*i*) a Generic Infusion Pump User Interface (GIP-UI) architecture is presented that can be used to reason about design defects in infusion pump user interface software that may potentially cause use hazards; (*ii*) an analysis of infusion pump incident reports and other information sources is presented that summarizes the common use hazards in infusion pumps on the market related to number entry tasks; and (*iii*) a preliminary hazard analysis that uses the GIP-UI to identify and reason about design errors commonly present in infusion pump number entry software and their relation to use hazards.

It is worth noting that the use hazards considered in this work, as well as software design defects that cause these hazards, are common in other interactive medical devices (e.g., ventilators) that have similar number entry systems. Therefore, we believe that the GIP-UI architecture may also be useful to the hazard analysis on these devices.

Organization. Section 2 presents background information about the GIP reference model. Section 3 elaborates the details of the GIP-UI architecture. Section 4 demonstrates how the GIP-UI architecture can be used to facilitate the analysis of use hazards related to user interface (in particular, the number entry software of a user interface). Section 5 compares our work with other related work. Finally, Section 6 concludes the paper.

2 Background: the Generic Infusion Pump model

The GIP model is a safety reference model for infusion pumps that captures the common functionalities of modern infusion pumps. Figure 2 illustrates an abstract architecture of the GIP model, and the functionalities of each of its components are follows:

The *GIP controller* represents an abstraction of the pump software that manages the overall infusion process and supervises interaction among all GIP components. It is responsible for instructing the pumping mechanism to deliver fluid as prescribed, as well as detecting and reporting alarm and warning conditions.

The *GIP user interface (GIP-UI)* represents an abstraction of the device elements (hardware and software) that enable interaction with the user. That is, the user can enter data (e.g., infusion parameters and pump settings) and control pump operation (e.g., start/stop infusion). The user can also receive from the user interface feedback on the pump and infusion status.

The *device data recorder* represents an abstraction of the logging mechanism used to record the pump's operational history (such as critical data and important events) to facilitate problem diagnosis.

The *fluid pathway* represents an abstraction of the following elements: fluid reservoir, which stores the fluid to be delivered; infusion set, which is usually an IV needle; and delivery interface, which is a tube that connects the fluid reservoir to the infusion set.

A more detailed description of the GIP model can be found in [10, 36].

3 GIP User Interface (GIP-UI)

The GIP model was originally designed to provide a basis for analyzing the safety and correctness of the software control logic in infusion pumps. It had a simple means for input/output, and therefore lacks any design or engineering details on user interface design. The GIP-UI architecture, depicted in Figure 1, replaces this user interface part with new architectural details, in order to establish a basis for reasoning about use hazards and interface design errors in infusion pumps.

One important principle in the GIP-UI architecture is to enforce a clear separation between high-level functionalities of user interactions and low-level functionalities that enable communication among its components (e.g., user interface elements that translate electrical signals into logical events). Moreover, issues associated with user manuals and use environment are also considered to enable a more comprehensive (hazard) analysis of user interface design.

The role of each component in the GIP-UI architecture is defined as follows.

GIP-UI widgets. This component represents an abstraction of the mechanical and hardware elements of user interface that enable pump-user interaction. Widgets can be either input or output widgets. Examples of input widgets include buttons, switches, and knobs, while examples of output widgets are displays, alarms, LED lights.

GIP-UI Software. This component represents an abstraction of the software regulating all functionalities of the user interface. It translates the user input received from the input widgets to commands understandable by the pump control software, manages interactive tasks such as number entry, and manages the output widgets to provide feedback (such as alarms and infusion status) to the user.

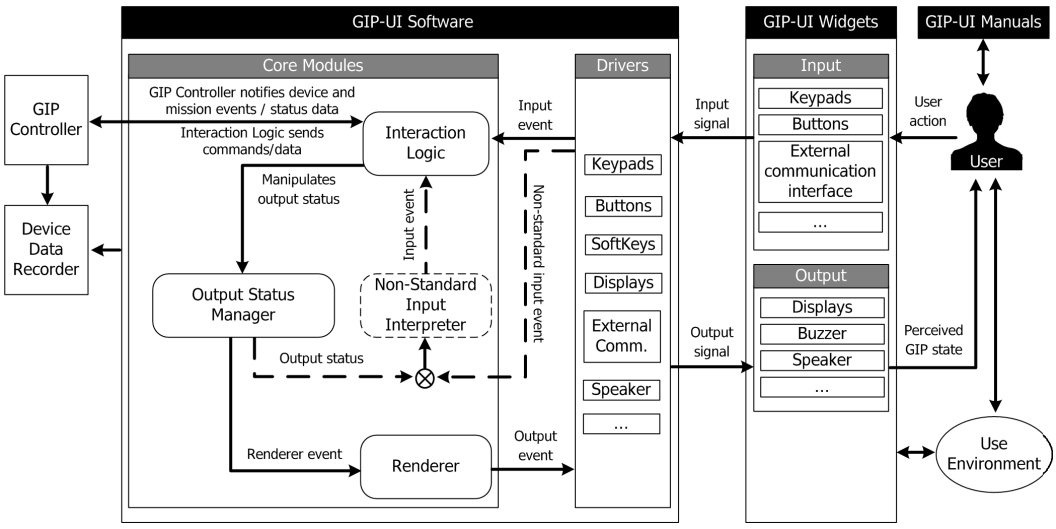

Figure 1 Logic architecture of the GIP user interface (GIP-UI). Labeled boxes represent an abstraction of functional components of the system. Arrows between components represent flows of information, user actions, or mechanical force between components. Dashed lines represent optional components and information flows.

The GIP-UI Software includes the following logic modules:

- *Core Modules.* These software modules define the interactive behavior of the device, i.e., how the pump responds to input events and how the device's operational state is updated as a consequence of input events. Input events are either from the user (labeled as 'user actions' in Figure 1) or from the pump control software (as abstracted as GIP Controller in Figure 2). The events from the GIP Controller are labeled as 'mission events' and 'status data' in Figure 1).

  The internal structure of the Core Modules is an instantiation of the well-known Model-View-Controller (MVC) [16] architectural pattern. The MVC pattern is chosen because it promotes a clear separation among different aspects of a user interface's core functionalities: defines the logic of interactions, decides what output information to be sent to the user, and decides how the output information is presented to the user. The Core Modules consist of:

  - The *Interaction Logic* , which represents an abstraction of the routines for handling user input events and forwarding them to the pump control software. These routines define: (i) input key sequences needed to interact with the device, and corresponding algorithms for processing input key sequences, and (ii) the communication protocol with the pump control software.
  - The *Output Status Manager* , an abstraction of the routines that specify feedback information, i.e., *what* feedback is presented to the user.
  - The *Renderer* , an abstraction of the routines for rendering feedback information on output widgets, i.e., *how* feedback is presented to the user.
  - The *Non-standard Input Interpreter* , which represents an abstraction of the routines for managing input widgets that are more complex than keypads or mechanical buttons. Examples of the non-standard input widgets include touchscreens or gesture recognition systems. For these input widgets, additional computation is needed to correctly interpret the user inputs. For example, in touchscreens, user gestures identify

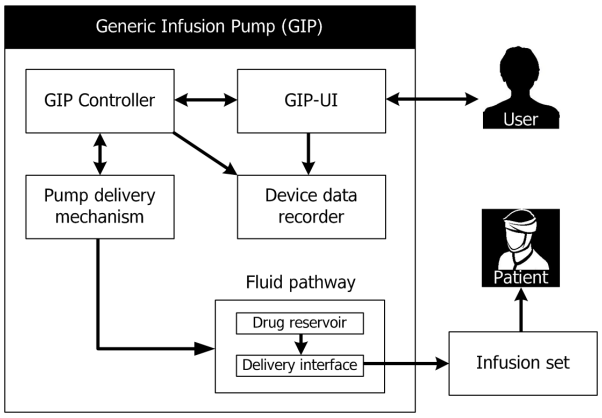

Figure 2 Logic architecture of the GIP. Labeled boxes represent an abstraction of functional components of the system, and arrows represent flows of information (e.g., software commands, hardware signals), user actions, or mechanical force between components. Note: Patient may overlap with User for some types of infusion pumps.

just coordinates on the screen, and further processing is needed to associate the screen coordinates to the user action (on the input widgets rendered on the screen)[1].

- *Drivers.* These software modules translate digital signals received from Input Widgets into events that can be processed by the GIP-UI Core Modules, or translate output events instructed by the Core Modules into output signals that Output Widgets can understand and produce output accordingly. The input events from the Drivers can be divided into two types: standard or non-standard. Standard events are generated during interactions with classical mechanical input devices such as mechanical buttons. An example of a standard event is a button click. Non-standard events, on the other hand, are generated during interactions with input elements such as soft buttons or voice recognition systems.

GIP-UI Manuals. This component represents an abstraction of the reference material accompanying the pump, e.g., user manual and training material. Such material is part of the user knowledge and is therefore useful for identifying potential use hazards.

User. This models the characteristics of the intended user population. Examples of user characteristics include: user training level, cognitive and physical abilities, attitudes and behaviors. Note that the user can be the patient or her family members for certain devices, and hence may not be trained nor be an experienced operator.

Use Environment. This component abstracts the physical environment in which the pump is used. Environmental factors such as ambient light conditions can affect the device's interaction with the user and thus should be included into considerations during user interface design. In Figure 1, arrows between the use environment and other components indicate what part of the system can be affected by the use environment. For example, inappropriate temperature levels may affect the physical abilities of patients and users, or go beyond the operating temperature of the device itself.

[1] See [6] for a detailed illustration of the function of this module.

4 Validation of the GIP-UI architecture

The GIP-UI architecture explicitly defines how the device interacts with the user. Thus, it can be used as a basis to examine where and how the device incorrectly interacts with the user, which may affect the device's expected operation and ultimately create potential hazards. This makes it easier to identify potential use hazards, or to cross-check the identified use hazards to ensure that no important hazards be missed.

More importantly, we believe:

> Having a generic GIP-UI architecture facilitates the identification of common design defects in user interface designs and provides insight into the cause-effect relationship between these defects and infusion pump use hazards.

To prove the above hypothesis, we conducted a preliminary hazard analysis (PHA) using the GIP-UI architecture to 1) identify a set of common use hazards in infusion pumps related to number entry tasks and 2) to reason about the common design errors in user interface software that may contribute to these use hazards.

4.1 Hazard analysis based on the GIP-UI

Our PHA focuses on the number entry part of the infusion pump interface. The *number entry software* in an infusion pump is responsible for managing interaction with the user when infusion parameters or pump settings need to be configured. This is chosen as the focus of our PHA because it is critical to infusion pump safety, in the sense that design errors in it can cause use errors (e.g., mis-programming of the pump) with potentially severe consequences to the patient (typically, overinfusion or underinfusion).

Our PHA of the number entry part follows a top-down approach: it starts from postulated undesired system outcomes, and then works out the causes of the postulated outcomes at the system design level. In the analysis, undesired system outcomes are given by use hazards documented in infusion pump incident reports, including FDA Adverse Event reports.

4.1.1 Scope of the analysis

Number entry software is designed to support a *number entry task*, which identifies the sequence of actions carried out by the user when entering infusion parameters or pump settings. In the current generation of infusion pumps, the typical number entry task is carried out through the following three main actions:

1. An infusion parameter or pump setting is selected by the user
2. The selected item is edited by the user
3. The value is submitted by the user

The actions described above generally involve pressing buttons and keys on the pump user interface. Whenever the pump registers a button press or a key press, the number entry software performs a computation. The computation may modify the device state (e.g., a new infusion rate may be configured in the pump), and generate feedback on the user interface to present the new device state to the user.

Buttons and keys currently used for number entry can be described using two broad classes of widgets: *serial number entry widgets*, which allow the user to enter the digits of the values serially from the most significant to the least significant digit, and *incremental number entry widgets*, which allow the user to modify an initial value by incrementing or decrementing it. Serial interfaces require a full numeric keypad, whereas incremental

interfaces typically have two or four arrow keys (depending on the exact style of interaction). A detailed description of these number entry widgets is in [3]. Notable for hazard analysis is how entry errors can be corrected. In a serial interface, either numbers have to be re-entered or there must be a *delete* key. In contrast, in incremental interfaces, the whole point of the user interface is to adjust numbers, so correcting errors does not require a separate *delete* key, as correction is just a special case of adjustment.

4.2 Sources of information

The analysis is informed by the following sources of information:

- Domain knowledge developed within the CHI+MED project (www.chi-med.ac.uk). This knowledge results from the analysis of commercial infusion pumps [11, 12, 19, 22, 23, 24, 27], infusion pump logs [17], incidents involving infusion pumps [13, 20, 21, 31], current and best clinical practice in hospitals and home care [29], and workshops with pump manufacturers, users and clinicians [4, 5];
- Adverse event reports collected through the FDA's Manufacturer and User Facility Device Experience (MAUDE) database [34];
- Recommendations on infusion pump design [25];
- Previous hazard analysis on other components of the GIP [10, 36];
- International standards ANSI/AAMI HE75:2009 on human factors, and ISO 14971:2007 on risk management.

4.3 Hazard analysis results

Using the GIP-UI our PHA identified 60 potential design errors in number entry software that may cause use hazards. From the GIP-UI perspective, these design errors arise from individual components in the architecture, from a combination of these components, or from the communication (i.e., information flow) between these components. The full table is in the technical report [28].

Table 1 is a sample of this hazard analysis table. The table shows the considered hazards at the top: over-infusion (or under-infusion) where the patient receives more (or less) drug/nutrients than prescribed. Each row in the table identifies a use error that can lead to the over-/under-infusion hazards, as well as an underlying design error in infusion pump software that can cause this use error. A concrete example (as found in incident reports or our previous study) is also presented for each identified issue, to help understand the nature of the corresponding issue.

For illustrative purposes, we present design errors found to be commonly present in number entry software of infusion pumps currently on the market. These design errors were found in real marketed devices and may lead to severe clinical consequences, as they can potentially lead to situations where the pump is erroneously programmed with incorrect infusion parameters or pump settings without the user's awareness.

4.3.1 Potential design errors in Output Status Manager

Design errors in the Output Status Manager generally lead to inappropriate, inaccurate, or incorrect feedback to the user. As a result, the user may not be able to receive adequate information on the system's status, what user action has actually been registered, and what result has been accomplished. Common instances of this type of design errors are as follows:

Table 1 Sample of the developed hazard analysis table. The full table is in the technical report [28].

Hazards: overinfusion or underinfusion		
Use error	**Underlying design error**	**Example**
The user fails to edit the value	The pump displays incorrect values without the user's awareness	The device shows the last programmed value instead of the current value
The user fails to select the intended input field	The pump displays instructions to the user prematurely due to incorrect assumption on user action	The device requires a rate value but the display shows a notification message "Enter VTBI"
The user fails to read values or units correctly	The pump uses inappropriate fonts or formats to render values or units	The device renders fractional values without a leading zero (e.g., .9 instead of 0.9), or integer values with leading zeros (e.g., 09 instead of 9)
The user fails to enter the correct digits	The pump erroneously discards key presses without the user's awareness	The device registers the key sequence 0 | . | 0 | 1 | without any warning or notification if the minimum legal rate is 0.1
...	...	...

- *Incorrect values are displayed without the user's awareness.* Pumps affected by this design error have an *ambiguous* display, i.e., the user is not able to tell the current system state from the observable information on the pump display. Consider the following error found in a marketed infusion pump. This pump displays the last programmed value or the last valid input value, instead of the current number entry value that the user enters. For example, it displays a value of 90 when the user starts the number entry task, since the last value programmed in the pump is 90. However, given that the user has not pressed any key yet, the current display value should be 0. This incorrect display value misleads the user in carrying out the rest of number entry. For instance, the user may press key 1 | in order to entering the number 901, while the actual value registered and displayed by the pump is 1. If not noticed, this can cause the user to incorrectly program the pump, and result in unexpected treatment to the patient. Our forensic analysis on this pump accredited the root cause of this error to initialization routines of the number entry system and routines for handling *illegal* input key sequences.
- *Instructions are displayed to the user prematurely due to incorrect assumption on user action.* Infusion pumps affected by this type of error usually display instructions that conflict with the ongoing actions taken by the user. A common cause of such errors is that the infusion pump incorporates routines that automatically validate the value entered by the user, even before the user actually finishes entering and confirms the value. Consider an instance of such errors found in a marketed infusion pump. The pump shows a notification message 'Enter Rate' when the user is expected to enter the rate value, and 'Enter VTBI' to instruct the user to enter the value of drug volume to be infused. However, if the user plans to enter a rate value of 109 ml/h but pauses for a

second after entering 1 | 0 |, the pump erroneously makes an assumption that the user has finished entering the rate, registers the entered value as 10, and displays the message 'Enter VTBI' to prompt the user to enter the drug volume to be infused. In other words, design errors of this kind may cause *mode confusion* to the user: the users thinks the pump is in mode x (in this case, x is entering the rate) while it is actually in mode y (in this case, y is entering the volume to be infused). In fact, this type of error ('right data, wrong field') appears to be the most common use errors documented in infusion pump incident reports [8].

4.3.2 Potential design errors in Renderer

Design errors in the Renderer module typically introduce visualization issues in the device. For example, the device renders information erroneously or inconsistently, or fails to make appropriate display elements perceptible.

- *Inappropriate fonts or inappropriate formats are used to render numbers or units.* Example problems found in marketed devices include: fractional values are displayed without a leading zero (e.g., displaying .9 instead of 0.9); integer values are rendered with leading zeros (e.g., 09 instead of 9); and small decimal point (e.g., the decimal point is rendered as · instead of •). Another example is with seven-segments displays [32]. Rendering values in seven-segment displays can easily cause the user to misread the values, as integer and fractional digits are hard to distinguish, and can result in out-by-ten errors.
- *Soft keys are incorrectly labeled without the user's awareness.* Soft keys are buttons that can be programmed to perform different functions during the pump's operation. Modern infusion pump usually implement soft keys as hard buttons placed on the sides of the screen, while soft button labels are displayed on the screen next to these hard buttons. We found it common for infusion pumps to display textual messages not intended to be soft button labels next to hard buttons, creating the illusion that they were. It is also common that infusion pumps erroneously render labels next to unused soft keys. Consider a pump with a soft key on the right of the screen, and another soft key on the left. Assume that these soft keys are aligned. If the pump renders 'Rate' aligned to the soft key on the left, and "2 ml/h" aligned to the soft key on the right, which soft key should be used to select the rate? The reasonable affordance[2] is that both soft keys be used for this purpose, as the two pieces of text are logically a single piece of information. The pump that we studied, however, disables one of these two keys, without providing any feedback when the user presses the disabled key. The likely clinical consequence of such errors is delay of treatment.

4.3.3 Potential design errors in Interaction Logic

Design errors in the Interaction Logic module generally result in incorrect human-computer interaction with buttons, keys, and displays. Typical causes of such errors include inappropriate or over-complicated procedures (i.e., sequence of actions) to interact with these widgets or the failure of implementing mechanisms for preventing or detecting user errors during interaction. Common instances of design errors in Interaction Logic include:

[2] Affordance is a property of objects that determines how the object can possibly be used [26].

- *Key presses by the user is erroneously discarded without the user's awareness.* Pumps affected may unexpectedly discard key presses and commit an out-by-ten error, which can potentially result in severe clinical consequences. For example, an infusion pump studied by us [3] enforces the constraint that all infusion rate values must be greater than or equal to 0.1. Thus, its number entry routines discards the third key press in the key sequence 0 . 0 1, as the value violates the constraint, and automatically sets the input rate as the minimum valuate 0.1. Moreover, the pump enforces that no fractional numbers are allowed for value above or equal to 100. Thus, the decimal point key press in the input sequence 1 0 0 . 1 is discarded, and the erroneous value 1001 is registered. No warning or notification is provided to the user in either of these two cases. The root cause of this design error typically resides in the number entry routines for handling illegal input values (e.g., out-of-range or ill-formed values).
- *Values entered by the user are erroneously discarded without the user's awareness.* Pumps affected by this design error unexpectedly discard the value entered by the user if the input field is de-selected (e.g., the user selects another input field without confirming this entered value). The following is an example detected in a marketed infusion pump: the user first changes the infusion rate from its current value 91 mL per hour to 0.9 mL per hour, and then selects the VTBI field to edit without confirming the new infusion rate. As a result, the pump automatically discarded the new rate value and maintained the previous value, without any notification or warning. Design errors like this can cause the user to mistakenly configure the treatment to the patient, and result in serious clinical consequences.

4.3.4 Potential design errors in Non-standard Input Interpreter

Design errors in the Non-standard Input Interpreter generally result in human-machine interaction issues with touch-screen displays. A device with such errors typically requires the user to take an inappropriate or over-complicated sequence of actions to interact with the non-standard input widgets. It may also fail to implement necessary mechanisms to prevent user errors during the interaction. Common design errors in the Non-standard Input Interpreter include:

- *Legal gestures on touchscreens are erroneously discarded without the user's awareness.* Devices with this erroneous design may ignore legal gestures on input widgets. For example, slide gestures on scroll bars are erroneously ignored; press and hold gestures on virtual buttons and tap gestures on input fields are erroneously ignored. This design error can be associated with number entry routines that activate or select touchscreen input widgets. The likely consequence of this kind of errors is the delay of number entry.
- *Similar gestures on touchscreens are erroneously associated with logically different functions.* Pumps affected by this design error erroneously execute functions not intended by the user. This design error may be associated with number entry routines that activate or select touchscreen input widgets, such as virtual buttons and input areas. The following is an example error detected in a marketed pump. The user selected the infusion rate field and plans to enter a value. However, the pump's touchscreen registers a tap gesture outside the area of the selected rate input field or outside the virtual buttons for number entry, probably because the user accidentally taps outside the widgets or the touchscreen

[3] More information about this study can be found in [24].

is mis-calibrated. Thus, the input field is automatically de-selected, without any warning or notification for the user. Notably, this design error may aggravate other design errors such as those in the Interaction Logic, as values may be mistakenly confirmed or discarded because of accidental touch on the touchscreen.

4.4 Discussion

As shown in the PHA results, the GIP-UI architecture provides a basis for us to enumerate common errors in user interface design associated with number entry. It also facilitates the establishment of the cause-effect relationship between these errors and the use hazards. As demonstrated in our PHA, this cause-effect relationship can help device developers make their hypotheses and assumptions explicit when reasoning about user interface behaviors, and thus enable clear thinking about possible causal factors. Ultimately, it can assist developers in designing effective mitigation measures to address use hazards and design errors causing them.

The GIP-UI architecture and hazard analysis results presented in this paper can be used by manufacturers as an independent reference to challenge the safety of their own user interface design. Alternatively, manufacturers can use our work as the starting point to perform more comprehensive hazard analysis on their products (in this case, manufacturers should populate the GIP-UI architecture with design details specific to their products).

It is worth pointing out that, even though our PHA identified a substantial set of use hazards and their root causes [4], it is by no means exhaustive. In fact, a PHA is only the first round of hazard analysis applied to a design, and its results constitute an initial yet informative inventory for subsequent detailed hazard analysis or the initial design of risk mitigation measures. In order to identify all potential hazards (and related causes) in complex systems, the best practice is to employ a combination of systematic hazard analysis techniques, such as Failure Mode and Effect Analysis (FMEA) [9] or Systems-Theoretic Accident Modeling and Processes (STPA) [18], in a complementary manner [14].

5 Related work

Although there are many examples of conceptual/abstract architectures that can be used for hazard analysis, few of them are designed to support the identification of use hazards and their potential causal causes in medical devices.

In [2, 1], an architecture is developed for a STPA-based hazard analysis for a radiotherapy system. The architecture describes the functions of five sub-systems of the Paul Scherrer Institute's experimental ProScan proton therapy system. Usability issues were considered in the study. However, this architecture was not designed to explore potential causal factors of use hazards, and some information on usability issues were obtained through workshops with domain experts. Similarly, in [30], a system architecture is developed for a STPA hazard analysis of pacemakers. Even though pacemakers do not have a user interface, the analysis considered a wider, system-level perspective that included users and patients. Users are modeled as 'human controllers' guided by mental models. Use hazards, however, were not in the scope of the analysis. The architecture presented in this paper can be used together with analysis techniques such as STPA. It provides information that can be used as a basis to

[4] The full hazard tables are in [28].

populate the *controller* and *controlled process* components of the system architecture with details necessary to analyze the potential causes of use hazards.

In [15], design errors related to barcoded medication administration systems are explored. These systems are commonly used together with infusion pumps and other medical devices to identify patients, clinicians, and drugs. The analysis in [15] revealed how defective barcoding systems encouraged workarounds that could potentially lead to severe clinical consequences. These results complement the results of our hazard analysis, in that they explore issues in the design of an alternative number entry system that can be installed on infusion pumps.

Other works studying use hazards in medical devices usually overlook design issues. For instance, in [35], training and clinical procedures are identified as potential causes of use hazards. Consider a use hazard where an incorrect patient profile is selected. The established causal relationship identifies nurses being unclear about the available profiles, and therefore better training is suggested as a mitigating measure. Even though training and clinical procedures may be contributing factors of use hazards, this approach to hazard analysis provides little or no insights about how an infusion pump can be (re-)designed to assure safety under existing training and clinical procedures.

6 Conclusions

A generic user interface architecture, GIP-UI, has been presented to facilitate the identification and reasoning of use hazards in infusion pumps. Its applicability to this end has been confirmed through a hazard analysis that involved known use hazards in marketed infusion pumps. The architecture was successfully used to reason about the cause-effect relations between these hazards and their causes (i.e., software design errors) commonly present in user interface designs.

The GIP-UI architecture can potentially be used as the basis for hazard analysis on use hazards, and for the establishment of safety requirements that ensure reasonable safety regarding device-user interaction. It is also worth noting that, even though the GIP-UI is designed for infusion pumps, the general idea behind it can be applied to other medical devices that rely on the same mechanism as defined in GIP-UI to interact with the users, and thus facilitate the assessment and assurance of their safety in device-user interaction.

Acknowledgment. This work is support by CHI+MED (Computer-Human Interaction for Medical Devices, EPSRC research grant [EP/G059063/1]). We thank Michael Harrison (Newcastle University) and Chris Vincent (University College London) for their valuable comments that helped us to improve our manuscript.

References

1 B. Antoine. *Systems Theoretic Hazard Analysis (STPA) applied to the risk review of complex systems: an example from the medical device industry.* PhD thesis, Massachusetts Institute of Technology, 2013.

2 A. Blandine, M. Rejzek, and C. Hilbes. Evaluation of stpa in the safety analysis of the gantry 2 proton radiation therapy system, 2012.

3 A. Cauchi, P. Curzon, P. Eslambolchilar, A. Gimblett, H. Huang, P. Lee, Y. Li, P. Masci, P. Oladimeji, R. Rukšėnas, and H. Thimbleby. Towards dependable number entry for medical devices. In *EICS4Med, 1st International Workshop on Engineering Interactive Computing Systems for Medicine and Health Care*. ACM Digital Library, 2011.

4 CHI+MED. Guidelines for number entry interface design (infusion devices). `http://www.chi-med.ac.uk/researchers/bibdetail.php?docID=684`, 2013.

5 CHI+MED. Personas and scenarios (infusion devices). `http://www.chi-med.ac.uk/researchers/bibdetail.php?docID=685`, 2013.

6 S. Conversy, E. Barboni, D. Navarre, and P. Palanque. Improving modularity of interactive software with the MDPC architecture. In *EICS2007*. ACM Digital Library, 2007.

7 ECRI Institute 2014 top 10 health technology hazards, 2014.

8 Association for the Advancement of Medical Instrumentation. Infusing patients safely: priority issues from the AAMI/FDA infusion device summit. `http://www.aami.org/publications/summits/AAMI_FDA_Summit_Report.pdf`, 2010.

9 P. L. Goddard. Software fmea techniques. In *Reliability and Maintainability Symposium, 2000. Proceedings. Annual*, pages 118–123. IEEE, 2000.

10 The Generic Patient Controlled Analgesia Pump Hazard Analysis. `http://rtg.cis.upenn.edu/gip-docs/Hazard_Analysis_GPCA.doc`.

11 M. D. Harrison, J. Campos, and P. Masci. Reusing models and properties in the analysis of similar interactive devices. *Innovations in Systems and Software Engineering, Springer-Verlag London*, 2013.

12 M. D. Harrison, P. Masci, J. C. Campos, and P. Curzon. Automated theorem proving for the systematic analysis of interactive systems. In *5th International Workshop on Formal Methods for Interactive Systems (FMIS2013)*, 2013.

13 ISMP Canada. Fluorouracil incident root cause analysis report. `http://www.ismp-canada.org/download/reports/FluorouracilIncidentMay2007.pdf`.

14 P. Jones, J. Jorgens III, A. R. Taylor 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.

15 R. Koppel, T. Wetterneck, J. L. Telles, and B. Karsh. Workarounds to barcode medication administration systems: their occurrences, causes, and threats to patient safety. *Journal of the American Medical Informatics Association*, 15(4):408–423, 2008.

16 G. E. Krasner and S. T. Pope. A description of the Model-View-Controller user interface paradigm in the Smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

17 P. Lee, F. Thompson, and H. Thimbleby. Analysis of infusion pump error logs and their significance for health care. *British Journal of Nursing*, 21(8):S12–S22, 2012.

18 N.G. Leveson. Software challenges in achieving space safety. *Journal of the British Interplanetary Society*, 2009.

19 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 *EICS2013, 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM Digital Library, 2013.

20 P. Masci and P. Curzon. Checking user-centred design principles in distributed cognition models: a case study in the healthcare domain. In *Proceedings of the 7th conference on Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society: information Quality in e-Health*, USAB'11, pages 95–108, Berlin, Heidelberg, 2011. Springer-Verlag.

21 P. Masci, H. Huang, P. Curzon, and M. D. Harrison. Using pvs to investigate incidents through the lens of distributed cognition. In Alwyn E. Goodloe and Suzette Person, editors, *Proceedings of the 4th NASA Formal Methods Symposium (NFM 2012)*, volume 7226, pages 273–278, Berlin, Heidelberg, April 2012. Springer-Verlag.

22 P. Masci, R. Rukšėnas, P. Oladimeji, A. Cauchi, A. Gimblett, Y. Li, P. Curzon, and H. Thimbleby. On formalising interactive number entry on infusion pumps. *ECEASST*, 45, 2011.

23 P. Masci, R. Rukšėnas, 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, Springer-Verlag London*, 2013.

24 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*. Springer Berlin Heidelberg, 2014.

25 National Patient Safety Agency (NHS/NPSA). Design for patient safety: A guide to the design of electronic infusion devices. `http://www.nrls.npsa.nhs.uk/resources/?EntryId45=68534`, March 2010.

26 D. A. Norman. *The design of everyday things*. Basic books, 2002.

27 P. Oladimeji, H. Thimbleby, and A. Cox. Number entry interfaces and their effects on error detection. In *Human-Computer Interaction – INTERACT 2011*, volume 6949 of *Lecture Notes in Computer Science*, pages 178–185. Springer Berlin Heidelberg, 2011.

28 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.

29 R. Rukšėnas, P. Curzon, A. Blandford, and J. Back. Combining human error verification and timing analysis: a case study on an infusion pump. *Formal Aspects of Computing*, pages 1–44, 2013.

30 Q. S. M. Song. *A system theoretic approach to design safety into medical device*. PhD thesis, Massachusetts Institute of Technology, 2012.

31 H. Thimbleby. Is it a dangerous prescription? *BCS Interfaces*, 84, 2010.

32 H. Thimbleby. Reasons to question seven segment displays. In *Proceedings ACM Conference on Computer-Human Interaction — CHI 2013*, pages 1431–1440. ACM, 2013.

33 US Food and Drug Administration, Center for Devices and Radiological Health (FDA/CRDH). *White Paper: Infusion Pump Improvement Initiative*, 2010.

34 US Food and Drug Administration (FDA). Manufacturer and User Facility Device Experience Database (MAUDE). `http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/PostmarketRequirements/ReportingAdverseEvents/ucm127891.htm`.

35 T. B. Wetterneck, K. A. Skibinski, T. L. Roberts, S. M. Kleppin, M. E. Schroeder, M. Enloe, S. S. Rough, A. S. Hundt, and P. Carayon. Using failure mode and effects analysis to plan implementation of smart IV pump technology. *American Journal of Health-System Pharmacy*, 63(16):1528–1538, 2006.

36 Y. Zhang, P. Jones, and R. Jetley. A hazard analysis for a generic insulin infusion pump. *Journal of diabetes science and technology*, 4(2):263, 2010.