Int. J. Human-Computer Studies 72 (2014) 284–297

Contents lists available at ScienceDirect

Int. J. Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

Analysing interactive devices based on information resource constraints ☆

José Creissac Campos [a,b], Gavin Doherty [c], Michael D. Harrison [d,e,*]

[a] Departamento de Informática, Universidade do Minho, Braga , Portugal
[b] HASLab/INESC TEC, Braga, Portugal
[c] Lero@TCD, School of Computer Science and Statistics, Trinity College Dublin, Ireland
[d] School of Electrical Engineering and Computer Science, Queen Mary University of London, London, United Kingdom
[e] School of Computing Science, Newcastle University, Newcastle upon Tyne, United Kingdom

ARTICLE INFO

Article history:
Received 13 February 2013
Received in revised form
1 August 2013
Accepted 2 October 2013
Communicated by Fabio Paterno
Available online 28 October 2013

Keywords:
Formal analysis
Task analysis
Distributed cognition
IV infusion pumps

ABSTRACT

Analysis of the usability of an interactive system requires both an understanding of how the system is to be used and a means of assessing the system against that understanding. Such analytic assessments are particularly important in safety-critical systems as latent vulnerabilities may exist which have negative consequences only in certain circumstances. Many existing approaches to assessment use tasks or scenarios to provide explicit representation of their understanding of use. These normative user behaviours have the advantage that they clarify assumptions about how the system will be used but have the disadvantage that they may exclude many plausible deviations from these norms. Assessments of how a design fails to support these user behaviours can be a matter of judgement based on individual experience rather than evidence. We present a systematic formal method for analysing interactive systems that is based on constraints rather than prescribed behaviour. These constraints capture precise assumptions about what information resources are used to perform action. These resources may either reside in the system itself or be external to the system. The approach is applied to two different medical device designs, comparing two infusion pumps currently in common use in hospitals. Comparison of the two devices is based on these resource assumptions to assess consistency of interaction within the design of each device.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

The process of assessing the usability of an interactive system is often criticised, either because it is biased by the expertise and judgement of the analyst or because it requires a sufficiently developed system to be able to assess it through user trials, either in the laboratory or in the "wild". These issues are particularly important when the system is safety or business critical. In such circumstances evaluation could make it necessary to make significant and potentially expensive changes late in the development process. User testing, while valuable, is also unlikely to cover all plausible user interactions with the system.

To explore usability, one starting point is to consider behaviours that achieve the intended goals of an activity (Butterworth et al., 1998). The focus of concern must be with what people might do with a device. This concern can be contrasted with a more complete analysis of every behaviour that an interactive device is capable of. Many of these behaviours, though undesirable, are unlikely to be carried out by a real user. Focusing on more likely user behaviour is often done by considering tasks or scenarios because they provide typical or intended behaviours. The problem is that what the designer intended or the scenario envisaged is not always how the system is actually used. Unexpected uses of the device can lead to entirely unforeseen usability issues. Our approach is based on a more situated view of interaction. It is assumed that interaction is shaped moment-by-moment by the information and affordances provided by the device, or the environment of the device. A typical interaction design issue that illustrates the role of resources is the "keyhole problem" in which users are distracted from achieving their primary goals by, for example, accessing different screens within a hierarchical menu structure in order to gather information that they require (Woods et al., 1994). The cues that are required to maintain an awareness

☆This paper has been recommended for acceptance by Fabio Paterno.

* Corresponding author at: School of Computing Science, Newcastle University, Newcastle upon Tyne, NE1 7RU UK.
Fax: +44 1912228232.
E-mail addresses: jose.campos@di.uminho.pt (J.C. Campos), Gavin.Doherty@tcd.ie (G. Doherty), michael.harrison@eecs.qmul.ac.uk, michael.harrison@ncl.ac.uk (M.D. Harrison).

1071-5819/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ijhcs.2013.10.005

of the main goal and their progress towards it are lost as the user navigates through the information space, viewing only a small proportion of the available information at a time. This places unreasonable demands on short-term memory and introduces vulnerability to a range of errors. Thus information needs provide constraints on user behaviour, and together with the resources afforded by the system, act to shape the likely behaviours of the end user. Users are therefore considered to behave by following paths that are suggested or enabled by *information resources*. Using them to drive analysis enables consideration of a broader class of uses and user behaviours than could be achieved by restricting analysis to the behaviours encoded in more prescriptive models. This makes it possible to explore many more behaviours than would be represented by, say, a task model (Kirwan and Ainsworth, 1992; Mori et al., 2002), and in some cases to realise that what the user might do is not consistent with what the designer had intended.

The proposed method involves specifying assumptions about information resources so that it is possible to check whether the right information is provided to users at the right time in support of activities. The designer must consider a range of issues when deciding which information is relevant at a given stage in the interaction: support for a range of user strategies, making the most of available screen space, avoiding information overload, and reconciling competing information requirements when the system supports a number of different activities. Subsequently, a model checker is used to find the more plausible behaviours within the space of all possible interactions. Traces that are generated by the analysis contain actions that are constrained by information resource assumptions. The model checking tool simplifies the process by automating the generation of these constrained behaviours. Each trace represents a scenario which is plausible with respect to the information resources and possibilities for action available to the user. Once these scenarios have been generated they can be explored by domain, software engineering, and human–computer interaction experts to consider their implications for design, and to decide whether remedial action needs to be taken (such as changing the device design). Some traces might be unexpected, perhaps bypassing some step which is important from a safety perspective, they might be longer than expected (as more efficient paths are insufficiently resourced), or there might not exist any well-resourced path to achieving the user's goal. The advantage of this approach is that it allows the analyst to focus on a subset of possible scenarios, ignoring implausible behaviours. It is always possible to model check the device without ignoring implausible interactions but this is likely to generate too many uninteresting behaviours, rendering this interdisciplinary analysis impractical.

This technique is designed to complement the systematic analysis of interactive devices using batteries of properties proposed by Campos and Harrison (2008, 2009) and Harrison et al. (2013). In the case of the systematic analysis no assumptions about use are made except insofar as they are captured in the properties themselves. For example, a property might state that a certain confirmation action is always performed, unless the user cancels the interaction. The property says nothing about whether the actions are sufficiently salient to be easy to use for example.

The paper extends work published by Campos and Doherty (2006) and Doherty et al. (2008). It compares two real systems that were both developed to support IV (intravenous) infusion in a hospital context. While the technique is intended to be generic to a range of modelling approaches it is illustrated using the IVY tool. This tool was initially developed to support the systematic analysis of interactive systems. The paper demonstrates a scaleable method for analysing interactive systems using constraints based on information resources, and to demonstrate the analysis of consistency properties and comparisons between different devices.

More specifically, the paper's contributions are:

- It demonstrates the use of *resources* as a modelling concept.
- It shows how resources can be used to focus analysis on plausible sequences.
- It illustrates the technique by contrasting the resources required in one real-world example with those in another both designed to support the same activities.
- The method also captures a number of different precise notions of task consistency and applies them.

The paper first discusses the background to this resource based approach (Section 2). Section 3 explains how resources can be used to support the analysis of systems in use. The manner in which resources are specified and the way in which goals are used in property formulation are discussed, along with the possibilities for tool support. This section introduces the proposed method. Models of the two infusion pumps are then briefly introduced in Section 4. Section 5 describes the activity context for describing the two devices before providing a discussion of the resource constraints relevant to the example (Section 6). The penultimate section uses the model of activities and description of the resources to compare the two models of devices (Section 7). Finally, discussion of the wider application of the method and of further developments is to be found in Section 8.

2. Background

The use of behavioural models, focusing on the system and supported by automated reasoning, to analyse human–computer interaction has been the subject of previous research (Mori et al., 2002; Campos and Harrison, 2001; Rushby, 2002; Loer, 2003). The particular tool that underpins the analysis performed in this paper is *model checking* (Clarke et al., 1999). Model checking is an automated approach that verifies that a formal model of a system satisfies a set of desired properties. It is a technique that is, in principle, feasible for use by non-experts given appropriate packaging of the mechanisms that are involved. One such mechanism is the use of generic models to describe classes of system (see, Garlan et al., 2003) that can be instantiated to particular examples. The infusion devices described in this paper use a common generic infusion pump specification that is used by the models of the interfaces to the two device models (Harrison et al., 2013). This eases the specification problem to some degree. The properties are typically expressed in temporal logic using the variables that describe the formal system to construct propositions. The method of proof is algorithmic which means that it is more accessible to non-experts than automated theorem proving. The appropriate formulation of properties and diagnosis when a property fails is not however a straightforward process. Property formulation can be made easier by offering general property templates (see, for example, Dwyer et al., 1999) that can be instantiated to the particular requirements of the devices. Property templates and the means of instantiation are offered by the IVY tool (Campos and Harrison, 2008).

When a property fails the model checker generates sequences of states (traces) in which the property being checked fails to hold. Each trace can be seen as representing a scenario. The scenario can be further explored by human factors or domain experts. However, not all behaviours that satisfy or fail to satisfy a property are of interest. Those that are of interest are the ones that are plausible because the device or the user's training, or some other contextual factor, lead to the particular behaviour. The problem is to ensure

that the interface helps the user to reach the goal and to avoid unsafe situations that might arise. While many features of human behaviour are unpredictable it is possible to determine systematic failures that are more likely to occur because of the way that the interactive system is designed. These behaviours are shaped by the users' goals and the system design, factors that are important if automated analysis is to focus on how the system design influences human behaviour.

Methods of folding the user into the analysis of system behaviour for realistic problems remains an issue. Existing research falls into two categories.

- It is concerned with analysis based on tasks, systematically exploring potential erroneous actions.
- It is concerned with forms of analysis that are based on a model of cognition.

2.1. Analysis based on tasks

The most recent example of this type of approach is the work of Bolton et al. (2012). Their work contains a good review of related material. It uses the Enhanced Operator Function Model (EOFM) to describe operator tasks. A task model is combined with a device model as a basis for a model checking analysis using SAL (de Moura, 2004). The analysis involves considering variants of the task by inserting automatically "phenotypes of erroneous human behaviour". These variant models are checked against correctness properties – that the combined model can reach specified goals. Observable manifestations of erroneous behaviour are also explicitly modelled by Fields (2001) who also analyses error patterns using model checking. Both approaches, however, whilst helpfully identifying specific kinds of errors to explore in the form of the mistake model, lack discrimination between random and systematic errors. They also assume implicitly that there is a correct plan, from which deviations are errors. A task based analysis technique, that has some connection with the information resource based approach described in this paper, analyses errors in terms of mismatches between the resources required by a task description and the resources provided by the device model, see for example Bottoni and Levialdi (2005).

Tasks are typically described in terms of the goal hierarchies and plans that are associated with intended or observed uses of the device. Task performance however often changes over time in practice. For internal or external reasons tasks may be performed in ways that were not originally envisaged. This is a particular problem when formal operating procedures based on tasks are designed to improve the safety of a system, for example users being expected to check that a particular value has been entered correctly before moving onto the next stage of the task. As an extreme example – consider a device in which no information at all is provided to the end user. Such a device could be seen as correct with respect to a task model (a user following the steps of the task would in theory achieve their goal). Similarly, vital information for performing a step in a task (e.g. displaying a drug dose so that the operator can check that it has been entered correctly) might not be provided by a device. Alternatively operators in practice might make use of emergent features of the device to perform tasks in ways not envisaged by the original designer.

2.2. Analysis based on cognition

An approach that avoids some of the difficulties of task models is to make assumptions about the cognitive behaviour of the user using a cognitive engineering perspective. For example *Cogtool* (Bellamy et al., 2011) uses ACTR to visualise the implications of design decisions by animating aspects of the design. This tool is primarily concerned with human performance given a specific user behaviour and is valuable in exploring these. Ruksenas et al. (2009) take a different approach. They explore the implications of a set of cognitive assumptions on a given design. The assumptions or principles are described formally in a generic user model (GUM). The approach checks properties of the combination of generic user model and device in order to generate paths that demonstrate flawed characteristics of the design from the perspective of these assumptions.

The assumptions about user behaviour made within these cognitive modelling approaches are stronger than those made for a resource based analysis, and involve a finer level of specification. The size of the generated models makes it difficult to explore detailed features of the device design in the way that is possible in our approach.

Taking inspiration from work on distributed cognition, a further step is to explore the distributed environment in which the system is situated which will in itself have a role in the cognitive tasks which are carried out. In this case it is recognised that a broader definition of resources beyond the device itself is required. A resource could be for example a post-it note reminding the user of a particular command that is useful in a particular situation. The DiCOT analysis method (Blandford and Furniss, 2006) is based on this view and has been formalised in Masci et al. (2012a) where the focus is incident analysis.

3. Resource based analysis

Resource based analysis takes user actions as the basic units of analysis. It combines these actions with resources that support them to define plausible interactions with a device, while avoiding the cost of developing a working user model. The process entails four phases (see Fig. 1). The first two involve producing a model of the device, and a model of the activities that the user will engage in to achieve the purpose of the device. The third phase is to describe the information resources that relate to the device's actions in the context of these activities. The fourth phase is concerned with analysis of the resulting models.

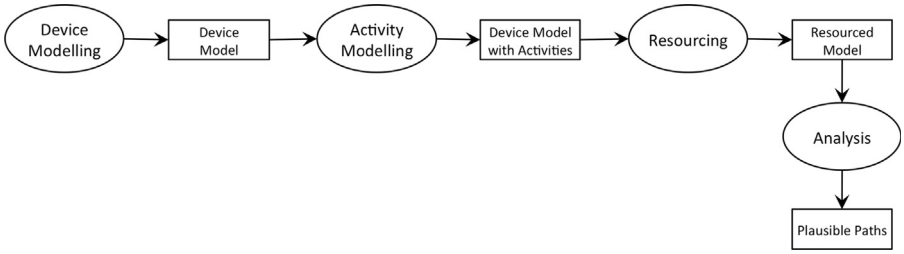


Fig. 1. Stages of the process.

3.1. Modelling the device

Interactive systems can be specified by defining the set of actions, including user actions, that are possible with them. These actions affect the state of the system and are determined by the mode of the device. A model of the device is proposed, and will be described in this paper, that focuses on interactive behaviour and makes explicit the relevant information resources. The focus will be on the types of analysis that can be performed and the nature of the results rather than providing full details of the specification. Further details of the modelling approach may be found in Harrison et al. (2013) and Campos (2012).

Modal Action Logic (MAL) interactors (Campos and Harrison, 2001) are used to describe the interactive behaviour of the device. Interactors enable the description of user interfaces as state machines. They consist of a set of proposition rules that aim to be more easily understood by designers and engineers, as discussed in Monk et al. (1991). Attributes are used to capture the information present in the user interface and user actions (as well as relevant internal state and actions – a distinction between both is achieved by assigning modalities to the former, e.g. vis). MAL is then used to write production rules that describe the effect of actions in the state of the device. The language also expresses when an action is allowed using permission axioms. MAL axioms will be used to define the behaviour of interactors.

Specifically, in addition to the usual propositional operators and actions the logic provides:

- a modal operator [_]_: [*ac*] *expr* is the value of *expr* after the occurrence of action *ac* – the modal operator is used to define the effect of actions;
- a special reference event []: [] *expr* is the value of *expr* in the initial state(s) – the reference event is used to define the initial state(s);
- a deontic operator *per*: *per*(*ac*) meaning action *ac* is permitted to happen next – the permission operator is used to control when actions might happen;
- a deontic operator *obl*: *obl*(*ac*) meaning action *ac* is obliged to happen some time in the future. Note that *obl* is not used in this paper.

Hence, the following extract of the model for a pump declares two boolean attributes (*poweredon* and *infusingstate*) and two actions (*start* and *pause*). It describes the effect of the action *pause* as setting an attribute *infusingstate* to false and leaving the attribute *poweredon* unchanged (priming is used to identify the value of the attribute after the action takes place), and restricts the *pause* action to only happen when the system is infusing and powered on. The *keep* function preserves the value of the attribute *poweredon* in the next state. If an attribute is not explicitly modified or is not in the *keep* list then its value in the next state is left unconstrained.

interactor *pump*
  attributes
    [vis] *poweredon*, *infusingstate*: *boolean*
  actions
    [vis] *start pause*
  axioms
    [pause] !*infusingstate*' $\wedge$ keep(*poweredon*)
    per(*pause*)$\rightarrow$*infusingstate* $\wedge$ *poweredon*

3.2. Modelling the activities

Assumptions about user activities are then added to the models. These activities achieve sub-goals, for example entering an infusion rate. Hence this part of the method involves breaking down the activities and goals that the device is designed to achieve. Information about the relevant activities can be obtained using established requirements elicitation techniques.

In terms of modelling, goals and subgoals are represented by (meta-)attributes in the model. These attributes do not correlate directly to the device, rather they represent the goal that the user is assumed to be pursuing in a given moment. This means that axioms for relevant device actions must be updated to consider which activities they support. Activities are represented by actions. These actions will act on the attributes introduced to represent goals, to model how activities achieve goals. The availability of actions and their effects are constrained by the state of the device (i.e. the attributes representing the device). Activities are related to device actions through the (meta-)attributes, representing goals that they manipulate. Hence in the example to be explored in later sections the attribute *phaserate* takes a value depending on what status the activity associated with entering the rate has reached.

3.3. Resourcing actions

The resourcing of each action is specified independently. The resourcing process addresses the information needed by the user to perform the action. Constraints (expressed as permission axioms) are specified over the attributes of the device and activity models. These constraints describe what information resources should be available for the action to be taken.

Resourcing will depend on the user's expertise with the system in a particular context. For example, if a mobile phone (the device) has an action to save a number to the address book, and this is the user's goal, then it is necessary that (1) action availability is resourced (the "save" option is currently on the screen), (2) the action is enabled (the device storage is not full), (3) action-effect information is available if the user is not expert with the phone (is the label "save to contacts" or just "save"?), and (4) required information about the current state is available (is it saved already?). Regardless of how a user accessed the number (is it in a list of missed calls or the content of a text message), or higher level user tasks and goals (which may be varied), the basic resourcing for this action remains much the same.

Wright et al. (2000) note that, in typical interfaces, information either in the device or in the environment can take specific forms:

- *Status/visible information* – perceivable information about the state of the device, for example a display indicating that a message is waiting, cues the user to perform an action to read the message. This is distinct from the system being in a state where reading a message is possible.
- *Action possibility* – indicating that an action is available, i.e. an action exists (e.g. the resource lets the user know they can save an unsent message for resending later, a feature they were unaware of); an action is enabled (or not) in the current state – perhaps the sender has no credit.
- *Action effect information*– indicating the likely effect of an action, for example "press ok to save" conveys information both on action possibility and on action effect.
- *Plan information* – that aids in the sequencing of user actions, for example, "You are in step 3 of 5".
- *Goal information* – helping the user to formulate and keep track of multiple goals, for example, "there are new messages".

These resources may be found in the physical context, in the device itself or in the user's head. The aim of the model is to make these resources generally explicit to understand how the design supports the user or otherwise. In practice, in the analysis described here, the information will be based on the device.

3.4. Analysis

Enhancing the specification with information resources enables a systematic consideration of the support they provide users. In the process of thinking about and specifying the resources, assumptions that are being made about the design will become explicit.

The model can then be used for further exploration of the device's behaviour. Typically this is achieved by attempting to prove desirable properties of the system, and making sure that undesirable properties do not hold. In the case of a resourced model, however, it becomes particularly interesting to attempt to prove that desirable properties will not hold (or that desirable states cannot be reached – e.g. that same activity cannot be carried out), as this will identify problems (if the property indeed does not hold), or provide traces that illustrate how, under the resourcing assumptions, the given property holds (the activity can be carried out).

Paths generated as a result of failing to satisfy properties will be paths that would have occurred if the user used the resources to support their interactions. It is possible then to investigate these paths and to construct scenarios around them or to explore properties of the interactions within these paths. The analysis then involves inspecting the generated traces that achieve the goals of the activities, and exploring the consistency of the devices in the context of these activities.

4. Modelling infusion pumps

Infusion pumps will be used to illustrate the approach. Because of the safety critical nature of infusion a number of agencies have produced sets of recommendations and safety requirements (US Food and Drug Administration, 2010). A more formal approach to analysis is likely to be more tolerable in this regulatory context. Particularly interesting and relevant to this paper is the generic infusion pump project underway within UPENN (Kim et al., 2011), CHI+MED (Masci et al., 2013) and the FDA. This project includes the definition of a number of safety requirements, including some that relate to the user interface of the device, although the focus is on device safety rather than user interface safety.

4.1. Infusion pumps

Infusion pumps are devices that are used mainly in hospitals to administer medication intravenously. They can be found in a variety of hospital contexts, including oncology and intensive care. Ambulatory versions can be carried by patients or used at home often for pain control. The users of these devices have varying levels of experience and training in how to use them. The setting up and running of an infusion requires the physical connection of the "giving set" from the pump to the patient's vein as well as programming the infusion device so that the correct volume will be infused over the prescribed time or at the prescribed infusion rate. This paper focuses on a particular activity that requires the programming of the pumps in question since the focus of the analysis is to explore the software design. Two pumps that provide similar facilities but have quite different user interfaces will be explored using information resource assumptions. For an approach, with some commonalities, that uses theorem proving combined with a distributed cognition based study of an oncology ward see Masci et al. (2012b).

Programming infusion pumps is an error-prone activity (see comments made in Ginsburg, 2005 for example) and therefore justifies the present focus. The two devices used for analysis in the paper have been modelled in detail and reflect a snapshot of two

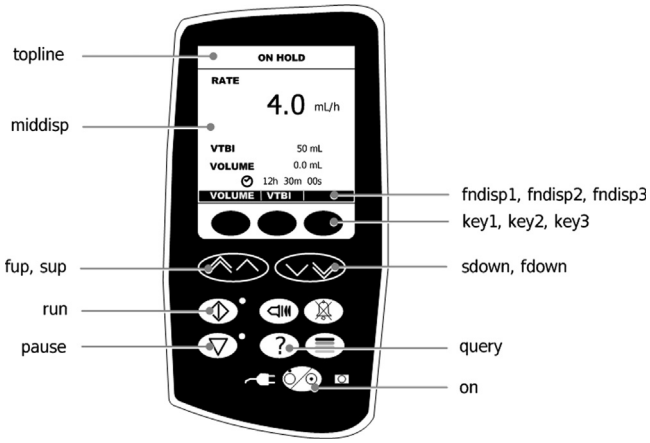


Fig. 2. The A pump (relevant attributes and actions).

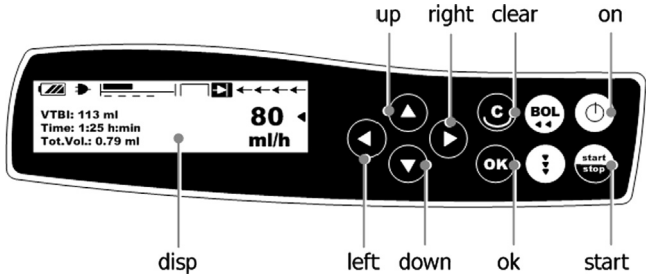


Fig. 3. The B pump (relevant attributes and actions).

systems that are undergoing continual evolution and improvement. Infusion pumps support a wide range of facilities and settings using the limited facilities of the keys and display by using modes. The first analysis is based on a device A, which is similar in detail to a version of the Alaris GP infusion pump (Cardinal Health Inc, 2006) (see Fig. 2). The second device B is similar to a version of the B. Braun Infusomat Space device (see Fig. 3). The models are detailed models but may not represent the current versions of the relevant marketed devices. These devices are typical pumps but have different user interface characteristics. These differences make them suitable for study and for comparison. For reasons of space the device models are only sketched here. Further details may be found in Harrison et al. (2013).[1]

4.2. Modelling the infusion pumps – the (basic) device model

Most infusion pumps have similar basic functions. They can be switched on and off, and they can be used to infuse medication intravenously using the pump. They can be used to program the pump parameters. Infusion pumps are designed to infuse a volume (vtbi – *v*olume *t*o *b*e *i*nfused) of medication intravenously subject to a prescribed infusion rate or over a prescribed time. The device recognises a variety of potentially hazardous conditions and uses alarms to alert the user. These conditions include the pump not having been switched off when the programmed volume has been infused, and the device having been left in hold mode without user action for too long. When the device is holding (that is paused), as opposed to infusing, the basic pump variables: infusion rate, vtbi, time and volume infused, can be modified although in some devices some of these variables can also be changed while infusing. Both pumps provide facilities for changing units and other parameters.

[1] The MAL models can be found at http://hicspecs.di.uminho.pt.

The basic underlying functionality is the same for most infusion pumps, therefore devices A and B have a common specification component that describes the pump's behaviour. The interface of each of the pumps is specified in separate components. The common component describes the behaviour of four actions *on*, *start*, *pause* and *tick*. The first three of these actions are user actions and turn the device on or off (*on*), start the infusion if the device is holding (*start*) or pause the infusion if it is infusing (*pause*). The *tick* action abstracts time and captures the infusion process: a relation between *vtbi*, *infusionrate*, *time* and *volumeinfused* as time evolves. When the device is infusing the vtbi is reduced according to the infusion rate every tick and added to the volume infused. *tick* computes the amount of time that has elapsed since the last action if holding. *tick* is invoked regularly and autonomously.

The pump interactor specifies attributes: *poweredon* (pump is switched on when true), *infusingstate* (pump infusing when true and holding when false), *infusionrate* (the rate of infusion), *volumeinfused* (how much has already been infused), *vtbi* (the volume to be infused), *time* (time to complete the infusion), *elapsedtime* (time since the infusion was started), *kvorate* (the keep vein open rate, which is a minimal rate that continues to be infused if the volume to be infused is exhausted), *kvoflag* (whether the pump is in keep vein open mode) and *elapse* (the time elapsed since the last key stroke when the device is holding). It also specifies internal actions that are only used by the main interactor to modify attributes. An example of such an action is *resetElapsed* that can be invoked by other interactors in the specification to set *elapsedtime* to zero.

4.3. The A interface model

Device A features a display and a number of buttons. Three areas of the display can be distinguished (see Fig. 2). A line at the top of the display presents status information, modelled by the attribute *topline*. For example, *topline*=*holding* indicates that the top line of the device, see Fig. 2, shows "ON HOLD". The middle part of the display, modelled by a boolean array *middisp*, displays the value of the pump variables and some other information. For example, *middisp*[*dvtbi*] indicates if true that the current value of *vtbi* is visible and the current value of *middisp*[*dquery*] indicates whether the options menu is visible. Finally, in the bottom part of the display, the labels over the three function keys are modelled by attributes *fndisp*1, *fndisp2* and *fndisp3*. Hence, *fndisp2*=*fvtbi* indicates that the display associated with key 2 shows "VTBI". The attribute names *topline* and *middisp* are not intended to indicate the spatial layout of the display. With hindsight it would have been preferable to use similar identifiers to those used in the BBraun model, that is *displaymode* and *disp*.

Other indicators are also defined. For example, *onlight*, *runlight* and *pauselight* are boolean attributes associated with lights on the device indicating whether the device is switched on, is running or is paused. It is clear that there is some redundancy between these displays and it is important to check that displays are consistent with respect to each other. Properties of this kind are discussed in Harrison et al. (2013).

The limited size of the display and number of buttons means that interface modes are inevitable. These modes are described using the state attribute *entrymode* that can take values:

- rmode – entry of the infusion rate via the chevron keys,
- bagmode – entry of vtbi by selecting from a menu of infusion bags,
- tbagmode – entry of vtbi by selecting from a menu of infusion bags when entering vtbi over time,
- qmode – selecting from a menu of options relating to a variety of device features,
- vtmode – entry of vtbi via the chevron keys,
- vttmode – entry of vtbi via the chevron keys when entering over time,
- ttmode – entry of time via the chevron keys,
- nullmode – variety of other situations when the chevron keys have no effect,
- infusemode – when the device is infusing.

This is captured in the type *emodes*:

$$emodes = \{rmode, bagmode, tbagmode, qmode, vmode, vttmode, ttmode, nullmode, infusemode\}$$

The A pump allows the user to move between modes using the three function keys (*key*1, *key*2 and *key*3). The meaning of the keys is indicated by the function key displays (as described above).

The pump supports number entry by using chevron buttons (modelled by actions *fup*, *sup*, *sdown*, *fdown*). These buttons are used to increase or decrease entered values incrementally according to mode as described above. The user will recognise the different meanings of the chevron key by using a variety of information.

- In infusing or holding mode when the top line is "OPTIONS" or "VTBI" and *entrymode* is *bagmode*. In this case the keys move the cursor up and down a menu.
- In holding mode when the top line is "ON HOLD", or "VTBI "or "VTBI/TIME" and *entrymode* is *rmode*, *vtmode*, *vttmode* or *ttmode* then the keys can change the value of infusion rate, vtbi or time depending on the combination.
- In infusing mode when the top line is "INFUSING" then the keys can change infusion rate.

4.4. The B interface layer

The B infusion pump differs in a number of ways from the A pump. It is driven by its menu interface. Entries in the menu allow changes to volume to be infused, infusion rate and time as well as other modifications similar to those provided by the A device's options menu. Number entry is achieved through a combination of cursor movement (using *left* and *right*) and incrementing/decrementing the digit pointed at by the cursor (using *up* and *down*).

The current mode of the device is determined by two attribute types: *dispmode* which indicates the displayed mode of the device and *emode* which defines the effect of *up*, *down*, *left* and *right*. The display modes:

$$dispmode = \{disprate, dispvtbi, disptime, mainmenu, dispinfusing, dispalarm, dispalarmvol, optionsmenu, statusmenu, dispblank\}$$

indicate the state of the menu at which the device has reached. For example *mainmenu* indicates that the device is showing the main menu, while *disptime* indicates that the device is showing time and is ready to enter a new value for time. The display modes are coupled with entry modes:

$$emode = \{dataentry, confirmmode, scalemode, nullmode\}$$

The entry modes determine the type of data entry. They have the following effect:

- dataentry – determines that numbers are to be entered, for example when coupled with a display mode of *disptime* then time is being entered.
- confirmmode – determines that the action is to be a confirm action, used in the case of an alarm display for example.

- scalemode – determines that entry changes values such as the volume of the alarm.
- nullmode – is a miscellaneous mode used when navigating menus for example.

As in the case of the A device a Boolean array *disp* indicates whether a pump variable is displayed.

5. Determining the activities

Understanding the particular role of a set of information resources in supporting usability in a design first requires assumptions about the activity context. The work of IV infusion can be described in terms of a collection of activities. These activities achieve sub-goals of the work, for example programming the infusion rate, confirming that the entered vtbi is correct or starting the infusion process. In the current case, these activities were elicited through analysis of operation manuals and interviews with clinical staff (Furniss, 2013).

These activities are described as actions. Unlike the actions described in the device model they describe how activity phases are affected by the activities. Activity actions also use device attributes, though they do not modify them. The device attributes used by the activities should be perceivable to users as they carry out their work. For example, the activity *entertime* is described by an action using two axioms: the first axiom defines when the action is permitted to occur and the second axiom describes the effect of the action. In the case of activities, the most important feature that indicates the constraints on the activity as a whole is when they are permitted to occur.

The activity *entertime* can only occur when time (*mtime* is non-zero) is specified as part of the prescription and when the vtbi, which is always the other prescribed value, is not currently being entered. The broader activity should be at a stage at which time is ready to be entered. These three constraints are expressed as

$$mtime\ !=0\ \&\ phasevtbi\ !=entering\ \&\ phasetime=ready$$

Further constraints involve information that is afforded by the device. They include assumptions about the information displayed or otherwise provided by the device. This information is assumed to be employed by the user when beginning the *entertime* activity. These device resources may trigger the commencement of the activity in the sense of an action possibility or may indicate to the user the status information that suggests readiness to engage in the activity.

In the case of the A device two features appear to provide status resources. They do not trigger commencement, rather they indicate that the activity may start. Indeed the user must perform other actions to arrive at this state as will be seen in the analysis (Section 7). The top line of the device (see Fig. 2) should show "VTBI/TIME" and in addition there should be an indication that time can be entered (this is indicated by an arrow which can only be seen in a particular mode of the device). These resource constraints are expressed by

$$topline=vtbitime\ \&\ entrymode=ttmode$$

Pulling these resources together the permission axiom in MAL is expressed as

$$per(entertime)\rightarrow(mtime!=0)\ \&\ (topline=vtbitime)$$
$$\&\ (entrymode=ttmode)\ \&\ (phasevtbi!=entering)$$
$$\&\ (phaserate!=entering)\ \&\ (phasetime=ready)$$

In the case of device B the display provides status information that is assumed to trigger the commencement of the activity. In this case there are a set of clearly distinguishable displays. The one that indicates that time can be entered is indicated in the B model by *displaymode*=*disptime*. The permission for *entertime* in the case of the B model is

$$per(entertime)\rightarrow(mtime!=0)\ \&\ (displaymode=disptime)\ \&$$
$$(phasevtbi!=entering)\ \&\ (phaserate!=entering)\ \&$$
$$(phasetime=ready)$$

Specifying what *entertime* does is independent of device details. The action simply specifies that the relevant phase of the activity (*phasetime*) becomes *entering*:

$$[entertime]\ phasetime'=entering\ \wedge \mathsf{keep}(\ldots)$$

6. Resourcing actions

The determination of activities involves a study of the work concerned with use of the device. The next step in the process is to consider the resources that are relevant to the device actions as indicated in Section 3.3. This involves identifying the resources and how they are combined to support this action. The resources are constraints on the actions and will typically involve a combination of device and activity constraints. These constraints are assumptions that describe the circumstances when an action is assumed to occur. The analysis considers the implications of these constraints, assuming that they are strict permissions. In this particular analysis all the assumptions relate only to the device itself or to meta-attributes relating to activities.. This process will be illustrated using two types of actions: the first type is used to enter data, the second type is used to change mode. The focus at this stage of the activity is to identify the resources that will be used to ensure that an action is carried out. The resourcing associated with an action is also described as a permission. These permissions further constrain behaviours that have already been expressed in the model of interactive behaviour described in Harrison et al. (2013). The permissions are added to capture how resource assumptions will further affect these behaviours. They then enable the exploration of the implications of these constraints.

6.1. Number entry actions

6.1.1. Resourcing data entry in A

In device A number entry is achieved using chevron keys. These have the effect of incrementing or decrementing the current value for a particular attribute (including the cursor position within a menu) either by a single unit (*sup, sdown*) or by a larger step (*fup, fdown*). The resourcing of these actions depends on what value is being entered (the activity that the user is engaged in) or which menu is involved. To demonstrate the process the *sup* action is considered. The key variables that can be entered using *sup* are vtbi, infusion rate and time as well as using the keys to navigate the options menu and a menu from which fluid bag volumes can be selected. The different alternatives are now considered. Two types of resource are important. The user's action is determined by the activity that they are engaged in and also determined by specific information communicated by the device.

6.1.2. Entering vtbi in A

Activity phase : *phasevtbi* should be *entering* (i.e. the user is entering vtbi). The operation *sup* is only relevant if the value of vtbi that has already been programmed into the device is less than the prescribed value of vtbi (*mvolume* is greater than *device.vtbi*).

Device resources : *vtbi* can be entered in two situations: when the top line of the A display shows "VTBI" and when it shows "VTBI/TIME". The user must realise a further piece of information in the second case, namely that the device entry mode (indicated by an arrow on the display) is set to enter vtbi, not time (modelled by *entrymode* = *vttmode*).

The information resources that constrain the use of *sup*, when entering vtbi, are described in the following specification fragment:

$$per(sup) \rightarrow (phasevtbi = entering \;\&\; ((topline = dispvtbi) | (topline = vtbitime \;\&\; entrymode = vttmode)) \;\&\; (mvolume > device.vtbi)) | \ldots$$

The axiom is completed by considering the other values that can be entered with the help of *sup*.

6.1.3. Entering time in A

Activity phase : In this situation the relevant phase attribute *phasetime* should be *entering*.

Device resources : The constraints in this case are similar to the vtbi case when the top line is "VTBI/TIME" but an arrow on the display indicates that the relevant mode *entrymode* = *ttmode* allows entry of time.

6.1.4. Entering infusion rate in A

Activity phase : In this situation the relevant activity phase *phaserate* should be *entering*.

Device resources : When the top line is "ON HOLD" the infusion rate can be entered. While it is also possible to change infusion rate when infusing, this option is not used as a constraint because it does not relate to the activities that have been determined to be important in this case.

6.1.5. Other situations when the chevron key is used in A

The chevron key may be used in relevant activities when either time or vtbi is ready to be entered but the process has not been started, or when entering *vtbi* and in the bags menu (vtbi can be programmed by selecting from a menu of predefined fluid bag volumes).

6.1.6. Resourcing data entry in B

In the case of device B, number entry is achieved by a combination of cursor movement (to modify the magnitude of the change) and incrementing or decrementing of the numeral at the position indicated by the cursor. The information resources associated with the B activities are derived in a similar way to the A device. As an illustration, consider the resourcing of the *up* key (the closest equivalent to *sup* in A) when entering vtbi (*phasevtbi* = *entering*). To reduce the search space for the model checker numeric values are modelled as binary numbers. The space reduction had no effect on the properties of the devices relevant to the analysis.

Whether *up* is appropriate for use in general depends on the difference between the currently displayed value and the prescribed value. Hence if the cursor is in position 3 (the least significant position in which an up or down action will have minimum effect) then the gap is required to be less than 2. In position 2, the gap is required to be between 2 and 4, and in position 1, between 4 and 8. A different approach is required than the case of device A to explore the details of the number entry. In this case the cursor, together with the value displayed and the value to be entered, provides an affordance that indicates whether the *up* action should be used.

In the case of entering vtbi, the value to be entered is *mvolume* and the display mode must be *dispvtbi*. This is captured by

$$(displaymode = dispvtbi \;\&\; phasevtbi = entering \;\&\; (entry = 3 \rightarrow mvolume - dispvalue < 2) \;\&\; (entry = 2 \rightarrow ((mvolume - dispvalue >= 2) \;\&\; (mvolume - dispvalue < 4))) \;\&\; (entry = 1 \rightarrow ((mvolume - dispvalue >= 4) \;\&\; (mvolume - dispvalue < 8))))$$

In addition to this there are other device resource constraints associated with whether the entry mode supports data entry. All this is captured by (only the resourcing for vtbi is shown):

$$per(up) \rightarrow (phaseinfuse! = entering \;\&\; (entrymode = dataentry \rightarrow ((displaymode = dispvtbi \;\&\; phasevtbi = entering \;\&\; (entry = 3 \rightarrow mvolume - dispvalue < 2) \;\&\; (entry = 2 \rightarrow (mvolume - dispvalue >= 2) \;\&\; (mvolume - dispvalue < 4))) \;\&\; (entry = 1 \rightarrow ((mvolume - dispvalue >= 4) \;\&\; (mvolume - dispvalue < 8)))) | \ldots)))$$

6.2. Mode change actions

This section considers actions in the two devices that cause the mode of the device to change. It is more difficult to provide direct comparisons between the two devices for these actions. However, there are a number of cases for A and B devices where *key*1 and *ok* act as confirmation keys. For this reason it is illustrative to provide a comparison between the resources for the two actions.

6.2.1. key1 in A

The chief role of *key*1 in A, in the context of the described activities, is to confirm the completion of the activity. The activity resources, required for completion of the activity when *key*1 would be used, are when the current activity is entering vtbi and when the vtbi set in the pump is equal to the prescribed volume

$$phasevtbi = entering \;\&\; device.vtbi = mvolume$$

The device resources relevant to this situation are either when the top line shows "VTBI/TIME" and the entry mode is appropriate, that is the arrow points at the correct value, or alternatively when top line is showing "VTBI". *key*1 is relevant in two possible modes in the latter case. The first is when the device value is the prescribed value (*device.vtbi* = *mvolume*) but the other is when the device is showing a bags menu (*middisp*[*dbags*]) and the selected menu entry shows the prescribed volume (*bagsval* [*bagscursor*] = *mvolume*). In each case the display associated with *key*1 should be "OK" (*fndisp*1 = *fok*). These characteristics can be brought together in the following:

$$per(key1) \rightarrow (fndisp1 = fok) \rightarrow ((phasevtbi = entering) \rightarrow$$

$$
\begin{aligned}
&(((device.vtbi = mvolume)\\
&\ \&\ ((topline = dispvtbi)\\
&\ |((topline = vtbitime)\ \&\ (entrymode = vttmode))))\\
&\ |((bagsval[bagscursor] = mvolume)\ \&\ middisp[dbags])\\
&\ )|\ldots)
\end{aligned}
$$

6.2.2. ok in B

ok is an action that has a similar effect for the device B. In this case it is assumed that if entering a pump variable then *ok* will be pressed if the value is the prescribed value. Likewise if the device is in the main menu and the phase relevant to the position of the cursor in the menu is *entering* and the relevant value is not equal to the prescribed value then *ok* will be pressed so that the user can start entering the value.

$$
\begin{aligned}
&per(ok) \rightarrow ((phasevtbi = entering)\ \&\\
&\quad (displaymode = dispvtbi)\ \&\ (dispinfvtbi = mvolume))\\
&\quad |((displaymode = mainmenu)\ \&\ (menucursor = dvtbi)\&\\
&\qquad (device.vtbi! = mvolume)\ \&\ (mvolume! = 0))\\
&\quad |\ldots
\end{aligned}
$$

7. Analysis

The previous two sections were concerned with:

- identifying activities associated with the work in which the infusion device is embedded;
- specifying resource assumptions associated with user actions required to achieve the goals of the activities.

Formulating the resource assumptions is itself a valuable activity. It can provide an understanding of the usability of the device, providing a walkthrough technique, similar to usability evaluation techniques such as cognitive walkthrough (Polson et al., 1992). It encourages the analyst to consider what information a user would need to help carry out each action. However the model can also be explored using a model checker. The value of model checking is that various forms of analysis can be performed. Two particular examples are now demonstrated.

1. This form of analysis involves generating paths that achieve activity goals consistent with the resource constraints. This process can lead to unexpected results, indicating surprising sequences that satisfy the resource assumptions but involve steps that were not anticipated. Paths achieving the same goals from the two devices provide an effective way of comparing them, for example comparing how intuitive the paths are in each case and whether there are actions that have no effective resource support.
2. This involves analysing the consistency of actions, particularly actions that cause activity or mode transitions.

Properties are presented for analysis using CTL (see Clarke et al., 1999 for an introduction to model checking). CTL provides two kinds of temporal operator, operators over paths and operators over states. Paths represent the possible future behaviours of the system. When p is a property expressed over paths, $A(p)$ expresses that p holds for all paths and $E(p)$ that p holds for at least one path. Operators are also provided over states. When q and s are properties over states, $G(q)$ expresses that q holds for all the states of the examined path; $F(q)$ that q holds for some states over the examined path; $X(q)$ expresses that q holds for the next state of the examined path; while $[qUs]$ means that q holds until s holds in the path. CTL allows a subset of the possible formulae that might arise from the combination of these operators. *AG(q)* means that q holds for all the states of all the paths; $AF(q)$ means that q holds for some states in all the paths; $EF(q)$ means that q holds for some states in some paths; $EG(q)$ means that q holds for all states in some paths; $AX(q)$ means that q holds in the next state of all paths; $EX(q)$ means that q holds in the next state of some paths; $A[qUs]$ means that q holds until some other property s holds in all paths; *E[qUs]* means there exists a path in which q holds until some property s.

7.1. Exploring the constraints associated with traces

The goal of using an infusion pump is to complete the prescribed infusion at a specified rate or within a specified time. This is achieved when the pump variable *volumeinfused* reaches the prescribed volume given that the rate and the time have been set up as prescribed and are not changed by the user during infusion. The clinician uses the pump to infuse the prescribed volume (*mvolume*) at a prescribed infusion rate (*mrate*) or over a prescribed time (*mtime*) to achieve the goal. Traces that capture sequences of actions required to achieve the activity goal can be determined by checking the property that this goal is never reached. By checking the negation of the property a trace will be produced that achieves the goal while at the same time satisfying all the resource constraints that have been asserted in the model.

The trace that is produced is examined to identify whether the paths are as anticipated, whether there are unanticipated actions that can occur that are not constrained by activity or device resources. Paths that achieve the activity goal can be explored using a CTL property that specifies that there exists no path which contains a state in which the volume infused is equal to the prescribed vtbi

$$AG(device.volumeinfused! = mvolume)$$

This property is checked for the two cases: where the infusion rate is supplied in the prescription, and where time is supplied in the prescription.

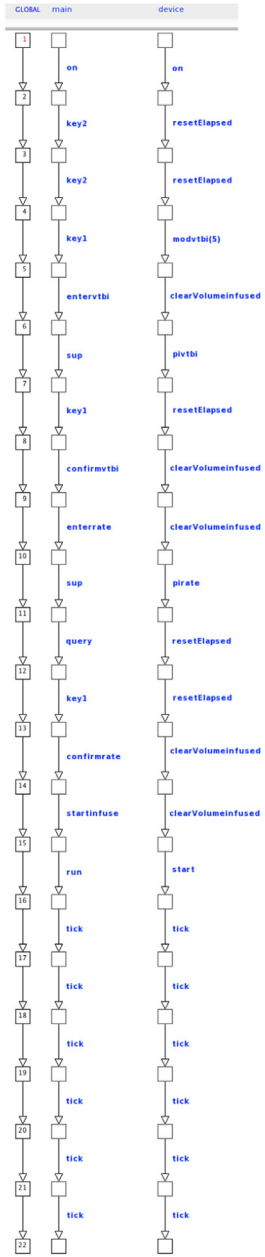
(i) $mvolume = 6$ and $mrate = 1$
(ii) $mvolume = 6$ and $mtime = 3$

These values are constants in the model that can be adjusted as required. The first involves entering vtbi and rate, the second vtbi and time. The values are tokens sufficient to reveal how the modes of the device are used to achieve the activity goals. The values are restricted because of the abstractions that have been used to ensure that the models are tractable. They do not affect an analysis of the mode structure of the two devices. The analysis explores the property above for the two cases across the two devices, which we denote as A(i) and A(ii) for device A and B(i) and B(ii) for device B.

The table described in the trace in Fig. 4 shows a failure of the property A(i) subject to the resource assumptions. Columns present the values of the attributes in consecutive states in the trace. Lines correspond to attributes, except for device.action and main.action which represent the action that has led to the state. The figure also describes the same sequence using a message sequence diagram. The IVY tool offers a number of alternative representations of traces. The trace can be summarised as follows.

- The pump is switched on (see main.action in state 2).
- Using *key2*, bag mode is entered and the bag at the default cursor position (a bag of size 5) selected (states 3-4). The user exits bag mode with *key*1 and goes into vtbi entry mode (state 5).
- The *entervtbi* activity is started (state 6). The *sup* key is used to reach the target vtbi (state 7).

a



b

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
device.action		on	resetElapse	resetElapse	modvtbi(5)	clearVolumi	pivtbi	resetElapse	clearVolum	clearVolum	pirate	resetElapse	resetElapse	clearVolum	clearVolum	start	tick	tick	tick	tick	tick	tick
elapse	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
elapsedtime	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6
infusingstate	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
infusionrate	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
infusionrateaux	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
kvoflag	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
kvorate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
poweredon	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
time	0	0	0	0	0	0	0	0	0	0	6	6	6	6	6	6	7	7	7	7	7	7
volumeinfused	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6
vtbi	0	0	0	0	5	5	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7
main.action		on	key2	key2	key1	entervtbi	sup	key1	confirmvtbi	enterrate	sup	query	key1	confirmrate	startinfuse	run	tick	tick	tick	tick	tick	tick
bagscursor	2	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bagsval[0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bagsval[1]	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
bagsval[2]	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
bagsval[3]	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
entrymode	nullmode	rmode	vtmode	bagmode	vtmode	vtmode	vtmode	rmode	rmode	rmode	rmode	qmode	rmode	rmode	rmode	infusemode	infusemode	infusemode	infusemode	infusemode	infusemode	infusemode
fndisp1	fnull	fvol	fok	fok	fok	fok	fok	fvol	fvol	fvol	fvol	fok	fvol	fvol	fvol	fvol	fvol	fvol	fvol	fvol	fvol	fvol
fndisp2	fnull	fvtbi	fbags	fnull	fbags	fbags	fbags	fvtbi	fvtbi	fvtbi	fvtbi	fnull	fvtbi	fvtbi	fvtbi	fvtbi	fvtbi	fvtbi	fvtbi	fvtbi	fvtbi	fvtbi
fndisp3	fnull	fnull	fquit	fback	fquit	fquit	fquit	fnull	fnull	fnull	fnull	fquit	fnull	fnull	fnull	fnull	fnull	fnull	fnull	fnull	fnull	fnull
middisp[0]	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
middisp[1]	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
middisp[2]	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
middisp[3]	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
middisp[4]	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
middisp[5]	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
middisp[6]	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
oldrate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
oldtime	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
oldvtbi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
onlight	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
pauselight	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
phaseinfuse	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	entering	entering	entering	entering	entering	entering	entering	entering
phaserate	ready	ready	ready	ready	ready	ready	ready	ready	ready	entering	entering	entering	entering	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed
phasetime	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready
phasevtbi	ready	ready	ready	ready	ready	entering	entering	entering	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed	confirmed
qcursor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rdisabled	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
rlock	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
runlight	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
topline	dispblank	holding	dispvtbi	dispvtbi	dispvtbi	dispvtbi	dispvtbi	holding	holding	holding	holding	options	holding	holding	holding	infusing	infusing	infusing	infusing	infusing	infusing	infusing

Fig. 4. (a) A state based representation of trace for A(i); (b) A tabular based representation for A(i).

- The vtbi is then confirmed (state 9).
- The activity (*enterrate*) is started (state 10). While the infusion rate phase is *entering* a single chevron up (*sup*) is used to set the infusion rate to 1 (state 11).
- The options menu is selected (via the query button) (state 12). *key*1 is pressed which locks the infusion rate (the first choice in the menu is to lock the infusion rate state 13). *key*1 returns the device to a top line of holding.
- The infusion rate is then confirmed (state 14).
- The *startinfuse* activity (state 15) leads to the run button being pressed. Infusion then continues until the volume infused is 6 (state 17–22).

This is one example of how a path where the property fails to hold illustrates, by negation, the goal being achieved. Before moving on to compare it with any other path, a number of issues should be noted. The shortest path to reaching the goal in this particular case is to use the menu options ("bags mode") to select a bag of size 5 and then to return to "manual" number entry to increment this bag size by 1. Because this particular path was not anticipated as part of *entervtbi* it is not constrained by an activity prior to entering vtbi as defined by the *entervtbi* activity. This will raise issues about what the clinician is expected to do – to follow a prescribed path or to be free to find the most efficient path.

The resources associated with selecting bags are relatively weak and yet the model checker takes this path because it is the shortest path. When thinking about the design of the device it seems likely that it was intended that bag mode would be used to select the exact bag sizes, rather than it being a quick way of reaching a particular value.

The second condition A(ii) produces a similar trace that describes the following path.

- *query* is invoked to access the menu entry for vtbi over time (*sdown*, *sdown*, *sdown*, *key*1).
- Once vtbi/time is selected the enter vtbi activity begins (*sup*, *fup*, *fup*, *sup*) and the value of vtbi is entered.
- *key*1 is pressed and has the effect of taking the device into *ttmode*, it is then possible to enter time.
- The entry of vtbi is confirmed (as 6).
- Time is entered (*sup*, *sup*) as 3.
- Time is confirmed and *key*1 is pressed which takes the device to a top line of holding.
- The infusion rate is locked (*query*, *key*1).
- Infusion is then commenced and the infusion completes.

Entering volume and time has a different mode structure than volume and infusion rate. The user must select the appropriate entry in the options menu and select vtbi and time as prompted by the device. The sequence of activities is reinforced by the way that the device works.

The difference between the mode structures of the two devices is evident when considering the same activities in relation to device B. Time, infusion rate and vtbi are all selected as modes from the main menu. The pattern of actions for the two modes of activity are very similar as can be seen in traces for B(i) and B(ii).

B(i) generates a trace indicating the following sequence of actions.

- Switch the device on.
- The entry of vtbi is selected from the main menu (*down*, *left*).
- The *entervtbi* activity is begun.
- *vtbi* is entered while *phasevtbi* is *entering* (*left*, *left*, *up*, *right*, *up*, *ok*).
- The *confirmvtbi* activity is completed.
- The *enterrate* activity is begun.
- While *phaserate* is *entering* the rate is entered (*up*, *ok*).
- The *confirmrate* activity is completed.
- The infusion process is started and completed.

An interesting feature of this device is that there are fewer alternative modes for entering the same information. There are therefore fewer ways to achieve the same goal. In the case of B(ii) the trace indicates the following sequence of actions.

- The device is switched on.
- The entry of vtbi is selected from the main menu (*down*, *left*).
- The *entervtbi* activity is begun.
- *vtbi* is entered (*left*, *left*, *up*, *right*, *up*, *ok*).
- The *confirmvtbi* activity is completed.
- The *entertime* activity is begun.
- The time is entered (*left*, *up*, *right*, *up*, *ok*).
- The *confirmtime* activity is completed.
- The infusion process is started and completed.

The two pairs of traces highlight differences between device A and B both in terms of number entry and their mode structure. Since number entry aspects of the model have been abstracted we focus on mode structure. All the traces considered satisfy the resourcing assumptions discussed in previous sections. In both cases A(i) and B(i) the move from the activity of entering vtbi to entering rate is marked by a confirmation action (*key*1 in A(i) and *ok* in B(i)). However although the same confirmation marks the transition from entering rate to starting the infusion in B(i) (*ok*), there is no such confirming action in A(i). Rather in A(i) there is an additional stage in which the infusion rate is locked prior to infusion (state 15). When the rate is entered in the A machine the transition to infusion requires no action that could be considered as a confirmation.

Furthermore in the case of A(ii) and B(ii) the differences between the two traces are more marked. The vtbi/time transaction is proceduralised in device A. The transition from entering vtbi to entering time is achieved through a single confirming action (*key*1). In the case of device B the entry of vtbi is confirmed (*ok*) and the next item in the menu that allows entry of time is selected by moving the cursor to the appropriate place. There is a regularity in B that is missing in A. However A provides an "accelerator" for the entry of vtbi over time.

Through this analysis of the sequence of steps it has become clear how the structure of the interactions between device A and device B differ for the particular instances of traces that achieve the two goals (i) and (ii). In the case of A the structure of the interaction reinforces the order in which the values should be entered and "forces" the user into a procedure. It also makes clear how the menu structure of B leads to a more straightforward selection of the appropriate mode for entering the different pump values.

7.2. Analysing properties

The previous section focussed on specific traces demonstrating how goals of the activity were achieved subject to resource constraints and other assumptions relating to the prescription used by the clinician. Alternative paths also achieving the same goals may also be considered by adding further constraints that exclude the possibility of generating traces already analysed. In this subsection more general analyses across all paths are considered. The focussing question is whether specific actions (or families of actions) consistently have a similar effect when making transitions between phases of the activity. For example, is the same action always used between ending one activity

and starting another when the user is constrained by resource assumptions?

This can be expressed as

$$AG(phasevtbi = entering \rightarrow AX(device.vtbi = mvolume \rightarrow !E[!confirmvtbiU(phase_x = entering)]))$$

It can be expressed either when $phase_x$ can be *phasetime* or *phaserate* depending on the initial prescription. The property requires that whatever path is taken after vtbi is entered then, before the next entry of a value, there must be a confirmation of the entry of vtbi. These properties relate to the generic activity of confirmation. In the case of A *key*1 provides the confirmation. For example,

$$AG(phasevtbi = entering \rightarrow AX(device.vtbi = mvolume \rightarrow !E[!key1U(phase_x = entering)]))$$

In the case of B, taking the example of entering time, prior to infusing:

$$AG(phasetime = entering \ \& \ entrymode = dataentry \rightarrow AX(dispinftime = mtime \rightarrow !E[!okU \ (phaseinfuse = entering)]))$$

For these properties to hold further constraints must be imposed on the actions that the user can take. None of the resource constraints specified so far prevents the user from taking unnecessary "detours" to perform the activities. The resource constraints do not prevent the user from taking some actions that may be discounted. However the fact that the properties fail provides valuable insights into unforeseen actions that could possibly be taken by the user. Discounted detours for analysis of confirmation actions include the following:

(a) While entering a variable, say vtbi, the user is assumed to remain in the relevant mode until completing entry of the value. This assumption is captured as a further constraint that prevents the possibility that the user will flip in and out of entering the relevant pump attribute. This constraint is expressed in both A and B as a MAL invariant:
In the case of A:

$$(phasevtbi = entering) \rightarrow (middisp[dvtbi] \ \& \ ((topline = dispvtbi)| \ ((topline = vtbitime) \ \& \ entrymode = vttmode)))$$

In the case of device B:

$$(phasevtbi = entering) \rightarrow (disp[dvtbi] \ \& \ (displaymodein \ \{dispvtbi, mainmenu\}))$$

The two assumptions reflect the different mode structures of the two devices. In the case of B *displaymode* of *mainmenu* is included because confirmation relates to the pump variable and not the temporary attribute that is used in *entrymode*.

(b) It is assumed that once the pump variable (again using the example vtbi) has been entered and confirmed the user will not change it until the infusing process changes it. This is captured in both A and B as

$$(phasevtbi = confirmed) \ \& \ (phaseinfuse = ready) \rightarrow mvolume = device.vtbi$$

(c) When the user starts the infusion, they will not change vtbi or any rate or time manually while the pump is achieving its goal. This is captured in both A and B as

$$(phaseinfuse = entering) \rightarrow ((device.vtbi + device.volumeinfused) = mvolume)$$

How reasonable these additional constraints are is a matter for the interdisciplinary analysis team to consider. Consideration of such constraints identified through the analysis might influence training, standard procedures, or further changes to the device design, such as issuing warnings or requiring additional confirmation actions from the user. The additional constraints are sufficient for the proof of both the confirmation action properties that were specified for device B. However it fails to be true in the case of device A for:

$$AG(phaserate = entering \rightarrow AX(device.infusionrate = mrate \rightarrow !E[!key1 \ U \ (phaseinfuse = entering)])$$

This property fails to be true because it is not necessary to type *key*1 to confirm entry of the infusion rate. It was noted in the trace analysis of the previous section that the A pump allows the user to move directly from entering infusion rate to commencing infusing without any confirming action taking place.

When checking an earlier version of the model the property relating to *phasevtbi*=*entering* was also found to be false. The generated trace indicated that the user might complete entering vtbi and be interrupted so that the device timed out thereby generating a warning. When clearing the alarm the device automatically returns to a state in which the topline shows "ON HOLD", leaving the vtbi mode. This change of mode would violate the property relating to the newly entered vtbi. Its preservation would not depend on the usual confirming action before moving to the next stage. Investigation of the behaviour of the physical device indicated that the device does leave vtbi mode, but as a quit action restoring the value of vtbi that the pump had on entry to vtbi mode. This loses work but maintains the requirement for an explicit confirmation action.

Resource constraints make explicit the conditions under which the interaction between the user and the device exhibits the desired behaviour using a similar approach to that described in Campos and Harrison (2001) and Rushby (2002). In the approach taken in this paper however the constraints are not encoded in the model but are separable from it.

In terms of the devices, the analyses of these properties makes clear the role of *key*1 with associated display *ok* in the case of device A, and in the case of device B the role of the action *ok*.

8. Conclusions

The resilience (or dependability) of a system does not depend only on the correctness of the system in terms of a set of predefined functional requirements. Folding the user into the analysis of a system's resilience will help to obtain a better understanding of how the system might operate in practice.

A resource based approach can help identify potential usability problems by exploring what should be available at the interface to support users. It also provides a means of comparing devices designed to support the same activities. The method can be applied iteratively, making changes to the system design, and exploring different activity assumptions. This paper has demonstrated that the availability of tool support not only makes it

possible to explore complex systems which are difficult to reason about reliably "by hand", but also allows the possibility of searching through the many possible behaviours for those which are of more interest to the analyst. The presence of resources introduces a notion of plausibility and a more realistic conception of user behaviour, which is based neither on rigid plan following nor random behaviour. User behaviour is shaped, in sometimes subtle ways, by system affordances and available resources, and by the user's own goals and strategies. Hence the properties that are analysed are concerned with the activities that the user carries out and how well they are supported, rather than analysing potential problems with the model. It is assumed that the model has also been evaluated for integrity and against a set of system properties as discussed in Harrison et al. (2013).

Other approaches fold human factors considerations into model-based approaches to usability reasoning as discussed in Section 2. The advantage of this approach over task related analysis as used by Bolton et al. (2012) is that the precise sequences are not specified. This allows the possibility of more exhaustive analysis of the paths that are plausible. While task-based analysis can be extremely valuable in design, consideration of safety properties should consider a wider class of plausible behaviours. The paper addresses difficulties relating to a task modelling approach by instead imposing constraints on behaviour. This narrows the analysis of interactive systems to *plausible paths*, that would be taken if assumptions about information resource constraints were valid, without limiting behaviour to specific sequences as suggested by the task representation. The information resource constraints will admit a greater number of paths than typically provided by a task model. It can be seen as consistent with Vicente's proposal in his critique of task modelling (Vicente, 1999) making allowance for the fact that users do not necessarily adhere to precise behaviours assumed by tasks. Users often achieve goals in ways that were not envisaged by designers when defining their task assumptions while at the same time recognising some constraints that will affect users. Some actions will be resourced effectively, others will not, leading to usability problems. While the resourcing analysis is informative in itself, the potential advantages of automated exploration of plausible behaviours are illustrated in Section 7.1, in which possible use of the bag mode as an accelerator emerges from the analysis, and in Section 7.2 in which attention is focussed on the impact of automatic mode changes; consistency is maintained at the cost of losing data entered by the user.

An alternative approach is to build an explicit model of the user. This is the approach taken in PUMA (Butterworth et al., 1998) and executable cognitive architectures (SOAR (Laird et al., 1987), EPIC (Kieras and Meyer, 1997), etc.). While these may deliver more detail in respect to cognitive and performance aspects, they are also more complex to construct in a form that is automatable (Ruksenas et al., 2009) and make stronger assumptions about how cognition works. Another approach is to encode assumptions about the user directly into the model (compare Rushby, 2002). In this case the separation between device model and user assumptions is less clear potentially and this can bias the user assumptions towards those that are needed to make the system work. By working with assumptions at a resource level, a clear separation is made between models and assumptions about users as expressed in terms of resources. These models are also relatively easy to build and the outputs are straightforward to work with, providing a natural way for the HCI expert to contribute to a more rigorous analysis.

The analysis presented complements a more unconstrained style of analysis where all possible behaviours of the system are analysed (which is useful when considering safety issues, for example). However, unlike many safety-oriented analyses, this approach is not about "proving" the system to be usable, but rather identifying and investigating plausible and interesting situations and behaviours to find and to fix usability problems (e.g. situations where insufficient resources are available to the user) and to investigate the effectiveness of different user strategies for achieving goals.

The infusion pump examples have shown the feasibility of the approach with realistically scaled devices. The analyses were run on laptop computers with now relatively standard quad core processors, and typically took a few minutes to complete. Development of the approach is continuing in a number of directions, including extending the approach to multiple users and making it easier to use by designers.

Acknowledgements

José Campos was funded by ERDF – European Regional Development Fund – through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - *Fundação para a Ciência e a Tecnologia* (Portuguese Foundation for Science and Technology) – within the project FCOMP-01-0124-FEDER-015095. Michael Harrison was partly funded by the CHI+MED project: Multidisciplinary Computer Human Interaction Research for the design and safe use of interactive medical devices project, UK EPSRC Grant number EP/G059063/1. Gavin Doherty would like to acknowledge the support of his research in part by Science Foundation Ireland Grant 10/CE/I1855.

References

BBraun, B. Braun Infusomat Space: Instructions for Use (Technical Report). B. Braun Melsungen AG. ⟨www.bbraun.com⟩.

Bellamy, R., John, B.E., Kogan, S., 2011. Deploying cogtool: integrating quantitative usability assessment into real-world software development. In: Proceeding of the 33rd International Conference on Software Engineering (ICSE '11), ACM Press, pp. 691–700.

Blandford, A., Furniss, D., 2006. Dicot: a methodology for applying distributed cognition to the design of team working systems. In: Gilroy, S.W., Harrison, M. D. (Eds.), Proceedings DSVIS 2005, Springer Lecture Notes in Computer Science, vol. 3941. Springer-Verlag, pp. 26–38.

Bolton, M.L., Bass, E.J., Sininiceanu, R.I., 2012. Generating phenotypical erroneous human behavior to evaluate human-automation interaction using model checking. Int. J. Hum.–Comput. Stud. 70, 888–906.

Bottoni, P., Levialdi, S., 2005. Resource-based models of visual interaction: understanding errors. In: IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 137–144.

Butterworth, R., Blandford, A., Duke, D., Young, R.M., 1998. Formal user models and methods for reasoning about interactive behaviour. In: J. Siddiqi, C. Roast (Eds.), Formal Aspects of the Human–Computer Interaction, SHU Press, pp. 176–192.

Campos, J.C., December 2012. Minho HCI Repository. ⟨http://hcispecs.di.uminho.pt.

Campos, J., Doherty, G., 2006. Supporting resource-based analysis of task information needs. In: Gilroy, S., Harrison, M. (Eds.), Interactive Systems: Design, Specification and Verification 12th International Workshop, DSVIS 2005 Newcastle upon Tyne, UK, July 2005, Springer Lecture Notes in Computer Science, vol. 3941. Springer-Verlag, pp. 188–200.

Campos, J.C., Harrison, M.D., 2001. Model checking interactor specifications. Autom. Software Eng. 8, 275–310.

Campos, J.C., Harrison, M.D., 2008. Systematic analysis of control panel interfaces using formal tools. In: N. Graham, P. Palanque (Eds.), Interactive Systems: Design, Specification and Verification, DSVIS '08, Springer Lecture Notes in Computer Science, vol. 5136. Springer-Verlag, 2008, pp. 72–85.

Campos, J.C., Harrison, M.D., 2009. Interaction engineering using the IVY tool. In: Calvary, G., Graham, T., Gray, P. (Eds.), Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems. ACM Press, pp. 35–44.

Cardinal Health Inc., 2006. Alaris GP Volumetric Pump: Directions for Use (Technical Report). Cardinal Health, 1180 Rolle, Switzerland.

Clarke, E.M., Grumberg, O., Peled, D.A., 1999. Model Checking. MIT Press.

de Moura, L., 2004. SAL: Tutorial (Technical Report). SRI International, Computer Science Laboratory, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA.

Doherty, G., Campos, J.C., Harrison, M.D., 2008. Resources for situated actions. In: Graham, N., Palanque, P. (Eds.), Interactive Systems: Design, Specification and Verification, DSVIS '08, Springer Lecture Notes in Computer Science, vol. 5136. Springer-Verlag, pp. 194–207.

Dwyer, M., Avrunin, G., Corbett, J., 1999. Patterns in property specifications for finite-state verification. In: 21st International Conference on Software Engineering, Los Angeles, California, pp. 411–420.

Fields, R.E., 2001. Analysis of Erroneous Actions in the Design of Critical Systems (Ph.D. thesis). Department of Computer Science, University of York, Heslington, York, YO10 5DD.

Furniss, D., 2013. Experiences of a HCI fieldworker on an oncology ward. In: Paper Presented at the Workshop on HCI Fieldwork in Healthcare. Creating a Guidebook, at CHI-2013. New York: ACM, Paris, France.

Garlan, D., Khersonsky, S., Kim, J.S., 2003. Model checking publish-subscribe systems. In: Proceedings of the 10th International SPIN Workshop on Model Checking of Software (SPIN03), Portland, Oregon, pp. 166–180.

Ginsburg, G., 2005. Human factors engineering: a tool for medical device evaluation in hospital procurement decision-making. J. Biomed. Inf. 38, 213–219.

Harrison, M., Campos, J., Masci, P., 2013. Reusing models and properties in the analysis of similar interactive devices. Innovations Syst. Software Eng. doi: http://dx.doi.org/10.1007/s11334-013-0201-3.

Kieras, D., Meyer, D., 1997. An overview of the epic architecture for cognition and performance with application to human–computer interaction. Hum.–Comput. Interact. 12, 391–438.

Kim, B., Ayoub, A., Sokolsky, O., Lee, I., Jones, P., Zhang, Y., Jetley, R., 2011. Safety-assured development of the GPCA infusion pump software. In: Proceedings of the 9th ACM International Conference on Embedded Software, EMSOFT '11, ACM, New York, NY, USA, 2011, pp. 155–164. doi:http://dx.doi.org/10.1145/2038642.2038667.

Kirwan, B., Ainsworth, L., 1992. A Guide to Task Analysis. Taylor and Francis.

Laird, J., Newell, A., Rosenbloom, P., 1987. SOAR: an architecture for general intelligence. Artif. Intell. 33, 1–64.

Loer, K., 2003. Model-based Automated Analysis for Dependable Interactive Systems (Ph.D. thesis). Department of Computer Science, University of York, UK.

Masci, P., Huang, H., Curzon, P., Harrison, M.D., 2012a. Using PVS to investigate incidents through the lens of distributed cognition. In: Goodloe, A., Person, S. (Eds.), NASA Formal Methods, Lecture Notes in Computer Science, vol. 7226. Springer, Berlin, Heidelberg, pp. 273–278, http://dx.doi.org/10.1007/978-3-642-28891-3_27.

Masci, P., Furniss, D., Curzon, P., Harrison, M.D., Blandford, A., 2012b. Supporting field investigators with PVS: a case study in the healthcare domain. In: Avgeriou, P. (Ed.), Software Engineering for Resilient Systems, Lecture Notes in Computer Science, vol. 7527. Springer, Berlin, Heidelberg, pp. 150–164 http://dx.doi.org/10.1007/978-3-642-33176-3_11.

Masci, P., Ayoub, A., Curzon, P., Harrison, M., Lee, I., Sokolsky, O., Thimbleby, H., 2013. Verification of Interactive Software for Medical devices: PCA infusion pumps and fda regulation as an example. In: Proceedings of the ACM Symposium on Engineering Interactive Systems (EICS 2013, ACM Press, pp. 81–90.

Monk, A., Curry, M., Wright, P., 1991. Why industry does not use the wonderful notations we researchers have given them to reason about their designs. In: Gilmore, D., Winder, R., Detienne, F. (Eds.), User-centred Requirements for Software Engineering. Springer, pp. 185–189.

Mori, G., Paternò, F., Santoro, C., 2002. CTTE: support for developing and analyzing task models for interactive system design. IEEE Trans. Software Eng. 28 (8), 797–813, http://dx.doi.org/10.1109/TSE.2002.1027801.

Polson, P.G., Lewis, C., Rieman, J., Wharton, C., 1992. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. Int. J. Man–Mach. Stud. 36 (5), 741–773.

Ruksenas, R., Back, J., Curzon, P., Blandford, A., 2009. Verification-guided modelling of salience and cognitive load. Formal Aspects Comput. 21, 541–569.

Rushby, J., 2002. Using model checking to help discover mode confusions and other automation surprises. Reliab. Eng. Syst. Saf. 75 (2), 167–177.

US Food and Drug Administration, Infusion Pump Improvement Initiative (Techncal Repoty). Center for Devices and Radiological Health (April 2010). ⟨http://www.fda.gov/MedicalDevices⟩.

Vicente, K., 1999. Cognitive Work Analysis. Lawrence Erlbaum Associates.

Woods, D.D., Johannesen, L.J., Cook, R.I., Sarter, N.B., 1994. Behind Human eError: Cognitive Systems, Computers, and Hindsight (Technical Report) SOAR 94-01. Crew Systems Ergonomics Information and Analysis Center (CSERIAC), Wright-Patterson Airforce Base, Ohio.

Wright, P., Fields, R., Harrison, M., 2000. Analyzing human–computer interaction as distributed cognition: the resources model. Hum.–Comput. Interact. 15 (1), 1–42.