chi+med
making medical devices safer

EPSRC Programme Grant EP/G059063/1

Public Paper no. 104

Reusing Models and Properties in the Analysis of Similar Interactive Devices

Michael D. Harrison, José C. Campos & Paolo Masci

Harrison, M. D., Campos, J. C., & Masci, P. (2015).
Reusing models and properties in the analysis of
similar interactive devices.
Innovations in System and Software Engineering, 11, 95–111.

PP release date: 30 January 2013

file: WP104.pdf

UCL UCLIC Swansea University Prifysgol Abertawe F I T L A B Queen Mary University of London CITY UNIVERSITY LONDON EPSRC Engineering and Physical Sciences Research Council

Noname manuscript No.
(will be inserted by the editor)

Reusing models and properties in the analysis of similar interactive devices

Michael D. Harrison · José Creissac Campos · Paolo Masci

Received: date / Accepted: date

Abstract The paper is concerned with the comparative analysis of interactive devices. It compares two devices by checking systematically a set of template properties that are designed to explore important interface characteristics. The two devices are designed to support similar tasks in a clinical setting. The devices differ as a result of judgements based on a range of considerations including software. Variations between designs are often relatively subtle and do not always become evident through even relatively thorough user testing. Notwithstanding their subtlety these differences may be important to the safety or usability of the device. The illustrated approach uses formal techniques to provide the analysis. This means that similar analysis can be applied systematically.

Keywords MAL · IVY · medical devices · procurement · interactive systems

1 Introduction and motivation

The systematic analysis of properties of interactive behaviour using Modal Action Logic (MAL) and the IVY tool has been reported in previous papers [10–12]. This paper uses these techniques focusing on the modelling and analysis of two devices developed by different manufacturers to support the same tasks. It builds on an earlier paper [12] that focused on the analysis of one of the devices described. This paper considers particular issues associated with reuse of elements of the device specifications and properties that are similar, while identifyig the ways in which they differ. The two devices manage limited displays and keys in different ways. A technique is demonstrated for comparing these user interfaces based on systematic analysis. The technique relies on the use of standard property templates. These devices are analysed, by reusing components of the specification that share common properties, and by systematically checking similar properties. The purpose of the analysis is to demonstrate:

- the use of a formal analysis technique as a means of comparison of the interfaces of two actual devices;
- particular mechanisms for the analysis based on the common use of components of the specification.

The approach will be demonstrated by producing analyses of versions of two intravenous infusion pumps, the Alaris GP [14] and the B.Braun Infusomat Space [3]. Although specific devices are being explored the purpose is to demonstrate the design and the judgements that can be made rather than to rate the two designs. These devices are used to indicate the scaling of the technique to real-world problems.

The contribution of this paper is to demonstrate the use of the IVY method and supporting tools in producing detailed specifications of the interactive behaviour of off-the-shelf state of the art devices. These specifications are approximately ten times the size of specifications produced before and demonstrate the scale-up of these techniques. The paper also shows how the analysis techniques allow an exploration of subtle issues relating to the interactive behaviour of the two devices,

Michael D. Harrison
School of Computing Science, Newcastle University E-mail: michael.harrison@ncl.ac.uk

José Creissac Campos
Departamento de Informática, Universidade do Minho and HASLab, INESC TEC

Michael D. Harrison, Paolo Masci
School of Electronic Engineering and Computer Science, Queen Mary University of London

and points towards a plausible method of comparison between interactive system designs.

Rigorous assessment is made particularly relevant by well documented concerns about the safety of infusion devices in general (see for example [24]). Failure to set up infusions accurately and reliably can have serious consequences. A number of incidents with a range of types of infusion pump indicate how the wrong material, or the wrong volume at the wrong rate, may in rare cases have disastrous consequences for the patient to whom the infusion was being administered (see, for example [42]). Part of the problem is due to vulnerabilities caused by interaction failure. Infusion pumps are designed to be used in a variety of settings, from general hospital wards, critical and intensive care, operating rooms to accident and emergency rooms. Some versions of the device are designed to be used at home or to be administered by the patients themselves.

The two pumps considered in the paper offer a number of facilities including number entry relating to the key infusion pump variables: volume to be infused, rate of infusion or time to infuse prior to starting the infusion process. The aim has been to produce specifications that are as accurate a snapshot of the devices as possible. However the purpose has not been to demonstrate in detail the characteristics of current versions but rather to demonstrate the utility of the proposed approach for exploring subtle design differences in a systematic way. Details of the description were derived from a combination of manuals ([14,3]), simulators, as well as some limited access to the physical devices themselves. The Alaris simulation is due to Patrick Oladimeji of Swansea University [33] based on the actual device and the other was provided by B.Braun of a version of the Infusomat Space. The approach to model elicitation is consistent with the "interaction walkthrough" suggestions contained in [40].

MAL and IVY were used to model the two devices. The models are illustrated by providing and explaining fragments of the MAL specification. These fragments both demonstrate process and reuse, indicating the scale of the undertaking that can be achieved using the MAL notation and the IVY tool. The paper demonstrates the extent of commonalities between the two specifications. It also demonstrates the common application of properties of interactive behaviour while at the same time showing that some properties were only relevant to one of the devices. A common layer specifies the attributes of the pump device, how the material is infused and at what rate or over what period of time. The paper briefly introduces specifications of the two devices. It focuses on properties that can be used for the purposes of comparison. After an overview of the background and related work (Section 2), the two infusion pumps are introduced (Section 3). Section 4 explains the three layers of the models and Section 5 provides examples of how generic properties were used in the verification of the model. Six main classes of properties will be considered: mirroring of device processes (e.g., infusing, on hold) in the interface; mode clarity; feedback for critical actions; consistency of the interface; checking ease of recovery; support for normative tasks. A final section (Section 6) provides some discussion of the comparison between the two devices and conclusions.

2 Background and related work

Techniques are required to help answer questions about the design relevant to safety both in the context of design and procurement. Safety in relation to *use* is an important aspect of this concern. In the specific context of procurement of medical devices a number of analysis techniques have been proposed, for example cognitive walkthrough [4], heuristic evaluation [42] and human reliability assessment techniques (SHERPA) [29]. The methods have not been adopted in practice because the cost of using them outweighs their perceived benefit. Often price is the overwhelming factor in deciding purchase. The manual and non-exhaustive nature of the techniques referenced above means that alternatives must be sought.

Also in the context of medical devices, empirical studies have been used to investigate how different design options may affect the reliability of user programming a medical device. Thimbleby and Oladimeji [34], for instance, used eye tracking to investigate the ability of users to detect errors on serial interfaces with a 12-key numeric keypad as compared to an incremental interface with chevron keys. Their results point out that incremental interfaces are likely to facilitate error detection. They argue that this is potentially linked to two factors. The first factor relates to the users' visual attention: for the incremental interface, users direct their visual attention towards the display most of the time, while in the serial interface they focus on the keys and usually omit checking the display. Because of this, device-dependent errors, like key bounces[1] due to defective buttons, are more likely to be detected when using the incremental interface. The second factor relates to syntax errors. With the serial interface, users may accidentally enter invalid numbers (e.g., 1.2.0 or 1..2), while such a possibility is designed out in incre-

[1] A key bounce occurs when physically pressing a button once causes a repeat of the same key.

mental interfaces. Invalid numbers need to be handled, and different manufactures generally implement different ad hoc solutions, with the net result that devices deliver an unpredictable user experience [38,39]. The work described in this paper complements empirical studies such as this one. It is concerned with subtle differences between apparently similar interfaces — aspects that would have been impractical to analyse in any experiment, and that are addressed by means of exhaustive analysis performed through formal verification techniques.

The broader analysis of safety critical systems using formal techniques has attracted considerable interest, and to some extent that has been extended to the analysis of safety issues associated with the use of these systems. Examples include those of Fields [23] and the more recent work of Bolton and others [6] who use model checking to explore the effect of user task deviations in a design. Other relevant research includes the use of finite state models to describe interactive systems, for example Thimbleby [41] and Heymann and Degani [25]. Reviews on the general topic of using formal verification to analyze interactive computing systems can be found in [13,7].

The IVY approach described in this paper have been applied to other examples including a simplification of a flight management system [9], an automobile based air conditioning system [10] and another simpler version of the devices analysed in this paper [11]. The analysis described in this paper is considerably larger than the previous work, the two models considered are each ten times the size of the models considered before. They provide detailed descriptions of the interactive behaviour of the two devices.

Other formal modelling techniques have also been applied to medical infusion pumps.

Bolton and Bass [5] used SAL [17] to analyse a model of the Baxter iPump. The pump model is developed within a framework that takes into account user goals, normative tasks (i.e., sequences of actions that must be performed according to written documents, such as user manuals or training material), and the operational environment. Their main goal is to explore the possibility of packaging an automated reasoning tool in such a way that non-experts in formal methods, such as human factors engineering practitioners, could (i) specify a realistic interactive system with intuitive modelling constructs, and (ii) verify in a reasonable amount of time a variety of basic normative tasks, such as turning on/off the pump, stopping the infusion, entering a volume to be infused. They performed the verification on a simplified model of the pump, as the state space of the full model exceeded the capabilities of the model checker. This work shares with that paper the concern that verification tools, when properly packaged, can enable non-experts of formal methods to perform the verification of realistic systems. In particular, a modelling architecture that enables model reuse can help to reduce the perceived cost of building a formal specification because representative models can be defined and easily changed to support multiple analyses. The work described in this paper differs from Bolton and Bass because its main goal is to compare similar designs of medical devices systematically. For this reason a detailed model of the interactive behaviour of the devices, limiting simplification to the minimum, is used. The layered approach described in the paper proves effective for this, as it allows some scope for reuse and it is possible to verify safety and reachability properties on the full specification of device modes. Examples of analyses will be explained in detail in section 5

Masci and others [31] analysed the interactive number entry systems of the B.Braun and Alaris pumps with the Symbolic Analysis Laboratory (SAL) [17]. They developed the specification by reverse-engineering the real devices from user manuals and manual exploration of the real devices. They formalised a design principle, *predictability* [19], which concerns the ability of a user to accurately predict the consequences of future interactions from the observable persistent state of the device (e.g., from what is shown on the device displays). Their verification approach systematically compares a *prediction model*, which specifies the expectations of the user, and relates it to the specification of the actual number entry system of the devices. The prediction model can be thought of as a mental model [26] developed by an idealised expert user that (i) knows perfectly the functionalities of the device, but (ii) makes decisions only on the basis of the current persistent state externalised by the device. Their analysis is conservative — if the ideal user fails to predict the effect of any action, then real human users would have similar difficulties (they can do no better than the idealised expert user). They are able to perform the analysis with SAL on detailed models that consider the full range of numbers handled by the real devices. This work complements the analysis described in this paper in that while it focuses on the details of the interactive number entry system while the present paper focuses on the device modes.

Also relevant is the work of Rukšėnas and Curzon [36] who are concerned to model not only the device but also to incorporate assumptions about the activities and goals that are to be carried out based on general cognitive principles. They have used their techniques to replicate experimental data relating to an ambulance dispatching system. This work goes beyond the current

paper by being concerned about cognitive processes although assumptions about infusion activities are made in this paper by specifying an "activity layer" (see also [20]) as is described in Section 4.4.

Finally, in the broader context of the verification of medical software where the user interface is not the focus, researchers at FDA and at the University of Pennsylvania have recently used a model-driven engineering approach to generate software for a prototype Patient-Controlled Analgesic infusion pump [27]. Starting from a Simulink Stateflow model, they translate the model manually into a specification using timed automata, verifying safety requirements with UPPAAL [30], and then synthesising C code with the TIMES tool [1]. Examples of safety requirements are: the pump shall issue an alert if paused for more than t minutes; each change in dose settings must be confirmed before it is applied. The complete list of the considered safety requirements can be found at [2]. The work described here complements these analyses by being concerned with interaction properties, such as mode clarity and consistency of naming and function.

Fig. 1 The Alaris GP Volumetric Pump (from [14])

Fig. 2 The B.Braun Infusomat Space Pump (from [3])

3 The infusion pumps

The aim here is to access relatively subtle properties of more complicated devices. The difference between this work and [12] is that the aim is to provide a means of comparison between two real devices both developed to support the same activity but in which, for a variety of reasons, quite different design decisions have been made.

The candidate devices for detailed analysis are the Alaris GP infusion pump [14] (see Figure 1) and the B.Braun Infusomat Space [3] (see Figure 2). In the following, the characteristics of the two pumps are illustrated along with the aims of the analysis.

3.1 Common features

Most infusion pumps have three basic states: infusing, holding and off. In the infusing state the volume to be infused (vtbi) is pumped into the patient intravenously according to the infusion rate. While in the infusing state the vtbi can be exhausted, in which case the pump continues in KVO (Keep Vein Open) mode and sets off an alarm. In holding state values and settings can be changed.

3.2 The Alaris Pump

When the Alaris pump is in holding state values and settings can be changed using a combination of function keys and chevron buttons (for the device layout, see Figure 3). A subset of the features that can be changed in holding state can also be changed when infusing. Number entry is achieved by means of chevron buttons. These buttons are used to increase or decrease entered numbers incrementally. Depending on current

mode the chevron buttons can be used to change infusion rate, volume to be infused and time, or alternatively allow the user to move between options in a menu, for example in bag mode and in query mode. Bag mode allows the user to select from a set of infusion bag options, thereby setting vtbi to a predetermined value. Query mode, which is invoked by pressing the query button, generates a menu of set-up options. These options depend on how the device is configured by the manufacturer, and include the means of locking the infusion rate, or disabling the locking of it, or setting vtbi and time rather than vtbi and infusion rate. There is also the possibility of changing the units of volume and infusion rate. The device allows movement between display modes via three function keys (*key1*, *key2* and *key3*). Each function key has a display associated with it indicating its present function.

3.3 The B.Braun Pump

The B.Braun pump provides a different mode structure. The different activities of entering vtbi, infusion rate and time can be accessed by means of a main menu that provides these activities as options. Values are entered for the relevant pump attribute by selecting the appropriate option and using four buttons. Two move the cursor left and right and two increment and decrement the digit associated with the cursor. When the number is *9* then *up* increments the value to *0* and carries, but the cursor remains in the same position and *down* has the opposite effect. There are fixed function keys: *clear*, *ok*, *run*. The availability of these fixed function keys is indicated in the current display.

3.4 The aims of the analysis

The focus of the analysis presented here is to consider confusions that arise as a result of the mode structures exhibited by these devices. Furthermore the analysis explores potential confusions relating to information that is being displayed. Properties fall into categories relating to:

- visibility of pump variables and therefore the process
- ease with which current interface mode can be discriminated
- feedback from actions
- consistency of action across modes
- ease of recovery from error
- resource support for the different activities for which the design is intended

The two devices above are compared in terms of "software" aspects of the design. The relationship between the device and its environment, which is of course extremely important, will not be considered explicitly in this paper. Hence, for example physical aspects of the "giving set" used to connect the device to the vein of the patient will not be considered nor will any issues associated with electromagnetic interference. These other aspects are also important in the design and procurement of these devices but are not the focus here.

There are a wide range of infusion pumps and syringe drivers in use in hospitals with a variety of interface styles that differ in terms of data entry and the ways in which modes are structured. This specific example will trigger discussion of broader issues.

4 The model

The specifications of the models discussed in this paper are available at the Minho HCI specification repository [8]. The model is described in three layers. Two layers are common between the two devices. The innermost "pump" layer captures the properties of the pumping device which is controlled by the user interface that is to be analysed. The layer is described by a reusable module which is instantiated in each specification. The middle layer is specific to the device being modelled and describes its interface structure. The outermost layer, the activity model, is described as part of the main module (to reduce state space overheads). This layer describes the constraints on the device that relate to the way it is used. It will assume the same activities in each device that is being considered. These activities will be derived from an understanding of the work that the clinician is to carry out. This outer layer is not dependent on characteristics of the device though it provides a mapping into the middle layer thereby grounding it in the specific details of the device.

4.1 The modelling language

The models are specified in a logic based on Structured MAL [37]. MAL (Modal Action Logic) is a (deontic) modal logic that incorporates a notion of *action*. Structured MAL adds mechanisms for structuring the specification to the basic MAL notation. In our case the notion of interactor [21,35] is used to structure the specification, borrowing the mechanisms from Structured MAL. Interactors are modules that have a state (defined by attributes) which is (partially) made available to the user through some presentation medium, and a set of

actions (some available to users, some internal) that act on that state.

MAL axioms will be used to define the behaviour of interactors. 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 these specifications.

One difference between the logic used here and Structured MAL is in the treatment of the modal operator. In Structured MAL the modal operator is applied to whole propositions. There is no way to relate old and new values of attributes directly. Old and new values are often related in practice by the introduction of auxiliary variables. For example an action (*tick*) which increments the value of attribute *elapsedtime* would be defined in Structured MAL as:

$$elapsedtime = aux \rightarrow [tick]\ (elapsedtime = aux + 1)$$

where *aux* is an auxiliary variable introduced to carry the value of *elapsedtime* into the next state (after *tick*). To avoid these auxiliary variables we extended the definition of the modal operator of [22] by using priming to state explicitly which references to attributes should be evaluated after the action. Hence the axiom above can be written as:

$$[tick]\ (elapsedtime' = elapsedtime + 1)$$

Parentheses will be omitted whenever the scope of the modal operator can be inferred.

The modal operator makes it possible to prescribe the effect of actions in the state but says nothing about when actions are permitted or required to happen. For this, permission and obligation operators must be used. As in [37], only the assertion of permissions and the denial of obligations are considered:

- $\mathsf{per}(ac) \rightarrow guard$ — action *ac* is permitted only if *guard* is true;
- $cond \rightarrow \mathsf{obl}(ac)$ — if *cond* is true then action *ac* becomes obligatory.

Permissions are asserted therefore by default and obligations are off by default. This makes it easier to add permissions and obligations incrementally when writing specifications. For example, the two permission axioms $\mathsf{per}(ac) \rightarrow guard1$ and $\mathsf{per}(ac) \rightarrow guard2$ together yield: $\mathsf{per}(ac) \rightarrow (guard1\ \&\ guard2)$ (note that & is used to denote logical *and* — | for logical *or*, and ! for *not*). This logic is particularly appropriate for describing a system in which components can be reused.

The interactor presentation is defined by annotating actions and attributes to show that they are perceivable. The modality of the perceivable attribute/action is given using further attributes. For example [*vis*] asserts that the attribute/action is visibly perceivable. In addition if attached to an action it can be invoked by the user. Additional annotations are introduced for further modalities.

Attributes and action parameters are typed. Types are represented as enumerations of the "key values" or as subranges of integer:

types
$T_{enum} = \{a, b, c\}$
$T_{range} = 0..10$

Interactors are composed through aggregation. For example the pump interactor can be assumed within the main interactor of the specification

interactor *main*
aggregates
pump **via** *device*

The three layers of the model are now described. More details of MAL can be found in [9–11].

4.2 The pump layer

The inner layer describes the basic infusion process. This process is captured in an invariant:

$$\begin{array}{lll} infusionrate > 0 & \rightarrow & infusionrateaux = infusionrate \\ infusionrate > 0 & \rightarrow & time = (vtbi/infusionrateaux) \\ infusionrate = 0 & \rightarrow & time = 0 \end{array} \quad (1)$$

This invariant asserts a relationship between vtbi, infusion rate and the time to completion of the process. *infusionrateaux* adds slight complication and is introduced to ensure that division by zero is avoided. It takes values in the range $1..maxrate$. The *tick* action captures the evolution of the process. It describes the steps in the process and the alarms that occur when the volume to be infused is exhausted, whereupon the device enters KVO mode, or when the device has been left in a hold state for too long. To illustrate the model's specification the normal infusion process is described using the MAL axiom.

$$(infusionstatus = infuse)\ \&\ (infusionrate < vtbi) \rightarrow [tick]\ vtbi' = vtbi - infusionrate\ \&\ elapsedtime' = elapsedtime + 1\ \&\ volumeinfused' = volumeinfused + infusionrate\ \&\ keep(kvorate, kvoflag, infusionrate, infusionstatus)$$

This axiom describes *tick* when the pump is infusing ($infusionstatus = infuse$) and *vtbi* exceeds the amount that is reduced as a consequence of the value of the infusion rate. The axiom has three elements: the action that is being described (contained in square brackets); the conditions that must be satisfied for the action to have the stated effect (left side of the implication); the result of the action under these conditions. $vtbi' = vtbi - infusionrate$ specifies that the next state (indicated by the prime symbol) of *vtbi* must be equal to its old value minus the *infusionrate*. MAL specifies that unless a state attribute is explicitly constrained in a modal axiom then it can change randomly in the next state. The *keep* function determines the list of state attributes that cannot change. The pump interactor involves 10 attributes, 3 actions (start, pause and tick) and 20 axioms.

4.3 The interface layer

4.3.1 Alaris

The difference between the two infusion pumps is captured in the middle interface layer of the specification. Both pumps use interface modes to make most effective use of the devices' limited display spaces. The middle interface layer describes the behaviour of interface modes. It also describes which of the pump variables are displayed, the displays associated with the function keys and the top line of the display which partly indicates the mode of the device. The Alaris display is organised into three parts. *topline* describes the contents of the top line represented by an enumeration of possible top line displays.

$$iline = \{holding, infusing, volume, dispvtbi, attention, vtbidone, dispkvo, setvtbi, locked, options, dispinfo, vtbitime, dispblank\}$$

middisp is a Boolean array that indicates whether a state attribute is visible. These state attributes are mainly, but not entirely, attributes specified in the underlying pump. Finally *fndisp1*, *fndisp2* and *fndisp3* represent the displays that describe the current purpose of the three soft keys. The following illustrative axiom describes the behaviour of the soft key 2 when the top line of the device shows either "holding" or "infusing" (see Figure 3) and the pump is processing normally (this is indicated when vtbi has not been exhausted which occurs when the pump is not in KVO mode, ie kvoflag is false).

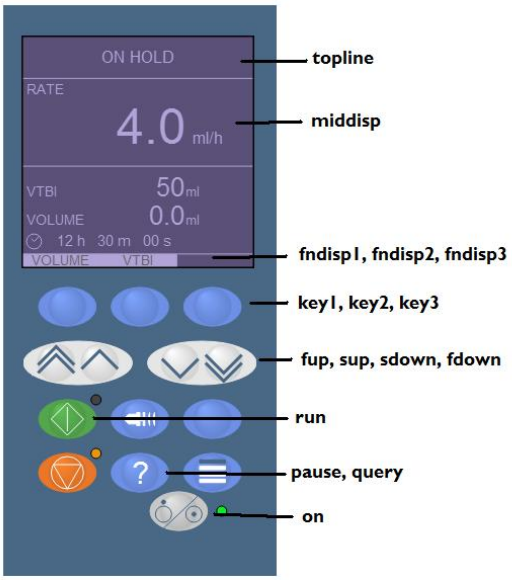

Fig. 3 Alaris actions(see [33])

$$(topline\ in\ \{holding, infusing\})\ \&\ !kvoflag \rightarrow [key2]\ topline' = dispvtbi\ \&\ oldvtbi' = vtbi\ \&\ middisp[dvtbi]'\ \&\ !middisp[dvol]'\ \&\ !middisp[dtime]'\ \&\ !middisp[dbags]'\ \&\ !middisp[dkvorate]'\ \&\ !middisp[dquery]'\ \&\ fndisp1' = fok\ \&\ fndisp2' = fbags\ \&\ fndisp3' = fquit\ \&\ entrymode' = vtmode\ \&\ effect(device.resetElapsed)\ \&\ keep(onlight, runlight, pauselight, rdisabled, rlock)$$

The axiom asserts that the effect of the action *key2* is that the next top line displays "volume to be infused" (*dispvtbi*) and the value of vtbi is displayed (i.e. $middisp[dvtbi]'$ is true) but other key pump variables and menus are not displayed. The soft function keys ($fndisp1'$ etc.) in this next step show "ok", "bags" and "quit" respectively. The mode specified by the value of *entrymode* is *vtmode*, which indicates that the device is ready to change the value of *vtbi*. Finally the elapsed time since there was last a key stroke when *infusionstatus* is *hold* is set to zero. *keep* indicates that the following state attributes are not changed by the action.

This Alaris interface layer consists of a specification that involves: 17 state attributes; 11 actions including augmentations of pump actions and 75 axioms.

4.3.2 B.Braun

The B.Braun layer has a different structure than that of the Alaris pump. The equivalent state attribute to *topline* is *displaymode* which specifies that there is information in the display that indicates the mode. *disp* is a Boolean array that represents the visibilities of state attributes including those that are specified in the pump interactor. There are actions *up*, *down*, *right*, *left* and *ok* that allow users to enter data. Hence the transition in the B.Braun pump that is equivalent to the Alaris transition that allows the user to begin entering vtbi is as follows:

$$
\begin{array}{l}
(displaymode = mainmenu)\&(menucursor = dvtbi) \\
\quad \rightarrow\ [ok]\ entrymode' = dataentry\ \& \\
\qquad displaymode' = dispvtbi\ \& \\
\qquad dispvalue' = device.vtbi\ \& \\
\qquad entry' = maxdigindex\ \&\ disp[dvtbi]'\ \& \\
\qquad !disp[drate]'\ \&\ !disp[dtime]'\ \& \\
\qquad !disp[detime]'\ \&\ !disp[dvol]'\ \& \\
\qquad !disp[dalarmvol]'\ \& \\
\qquad effect(device.resetElapsed)\ \& \\
\qquad keep(target, alarmvolume)
\end{array}
$$

This axiom specifies that if *displaymode* shows the main menu and the menu cursor is pointing at the vtbi entry then on pressing *ok* the device moves to data entry mode, as specified by *entrymode* and the *displaymode* shows vtbi, and the temporary state attribute used to show the number entered is set to the current value of vtbi. Only vtbi is shown on the display.

This B.Braun middle layer consists of a specification that involves: 16 state attributes; 9 actions including 3 augmentations of pump actions and 58 axioms. It should be noted that some of the possible set up options that are also provided by the B.Braun main menu are omitted from this model but the equivalent features are included in the Alaris model.

4.4 The activity layer

The third layer of the model describes the activities that are assumed to be typical of the clinician's use of the device. Whereas the pump layer, which is reused between devices, is specified as a separate interactor, the activity layer which has the same shape but different implementation in the two models, is in each case part of the interface interactor for the two devices. Activities are determined through negotiation with domain and human factors experts. The process of eliciting and understanding these activity actions and meta-state attributes is a process akin to task modelling [28] where the human factors or domain expert observes the infusion activity in context. What is described in the activity model differs from a task model because no assumptions are made about the user's plan. Instead the activity model defines constraints on activity that can lead to a variety of paths. The activity model described here is illustrative. No specific attempt has been made to observe clinicians as they set up infusions in their work context. One activity that may have been determined as being important in the infusion process for example is that another clinician should confirm that the correct value of vtbi has been entered into the device based on the original prescription. These meta-state attributes do not capture any feature of the device or its interface, rather they indicate a state in the activity as assumed to be understood by the clinician who is using the device. They can be used to constrain the device actions described in the interface layer. Hence confirming that the vtbi has been entered involves completing the *entervtbi* activity (by requiring that the meta-state attribute *phasevtbi* is set to *confirmed*).

$$
\begin{array}{l}
[confirmvtbi]\ phasevtbi' = confirmed\ \& \\
keep(\ldots, phasetime, phaserate, phaseinfuse)
\end{array}
$$

The *confirmvtbi* activity is only permitted if *phasevtbi* is *entering* and and the value of vtbi is equal to the required volume contained in the prescription. This activity is generic in the sense that nothing in its description depends on the Alaris or B.Braun interface. This is expressed in MAL as:

$$
\begin{array}{l}
per(confirmvtbi)\ \rightarrow\ (mvolume = vtbi)\ \& \\
phasevtbi = entering
\end{array}
$$

These meta-attributes are used to "resource" the actions [20] described in the interface layer. In the case of the Alaris pump the chevron key is permitted only when entering the vtbi or entering the infusion rate as part of an activity defined at the outer layer. These actions are now assumed to bear a specific relation to the prescribed value that the clinician has taken from the prescription. Hence the "fast up" chevron is only used if the current value is less than the required value by more than "big step" in the appropriate activity. This is specified using the following permission axiom.

$$
\begin{array}{l}
per(fup) \rightarrow \\
\quad (phasevtbi = entering\ \& \\
\qquad ((topline = dispvtbi\ \&\ middisp[dvtbi])\ | \\
\qquad\ (topline = vtbitime\ \&\ entrymode = vttmode))\ \& \\
\qquad ((bigstep + device.vtbi)\ <\ mvolume))\ | \\
\quad (phasetime = entering\ \& \\
\qquad (topline = vtbitime\ \&\ entrymode = ttmode)\ \& \\
\qquad ((device.time + bigstep) < mtime))\ | \\
\quad (phaserate = entering\ \&
\end{array}
$$

$$(topline = holding \ \& \ entrymode = rmode) \ \&$$
$$((device.infusionrate + bigstep) \ < \ mrate)) \ |$$
$$((phasetime = ready \mid phasevtbi = ready) \ \&$$
$$topline = options)$$

This permission axiom provides all the circumstances in which the button may need to be used and expresses them as constraints on the use of the button. For example, during the phase of the activity in which vtbi is being entered the top line of the device should either show "vtbi" and the vtbi should be visible or "vtbi over time" in *vttmode*, that is the cursor indicating mode should be in the upper part of the display. The permission also requires that the distance between the current value of vtbi and the prescribed value of vtbi should exceed the step that is made by the fast chevron key. These constraints make assumptions about activities, for example they assume that the clinician does not use "bag mode" to select the correct value of vtbi. This is the kind of assumption that is outside the province of the analyst and requires input from the domain or human factors specialist. This particular permission also describes the constraints that apply when entering time or entering infusion rate, and also allows use of the key to select appropriate options. Implementing the constraints for the B.Braun demonstrates differences between the two devices. The B.Braun has two separate functions that together allow number entry. Modifying the size of the increment is achieved using the *left* or *right* button. The decrement or increment is achieved using the *down* or *up* buttons. Number entry is only permitted if the device is in *dataentry* mode. Entering vtbi is only allowed, for example, if the display mode is to display vtbi, and the prescribed vtbi is greater than the currently displayed value of vtbi. The permission also includes similar constraints for *up* when the display mode is to display time or infusion rate.

$$per(up) \ \rightarrow$$
$$(entrymode = dataentry \ \rightarrow$$
$$(displaymode = dispvtbi \ \&$$
$$mvolume \ > \ dispvalue) \ |$$
$$(displaymode = disptime \ \&$$
$$mtime \ > \ dispvalue) \ |$$
$$(displaymode = disprate \ \&$$
$$mrate \ > \ dispvalue))$$

The activity layer captures some of the properties of the cognitive models described by [36]. It is however simpler, making no assumptions about the cognitive process itself, simply concerning itself with observed activities. This activity layer consists of a specification that involves: 4 state attributes; 7 actions (activities) and 21 axioms.

5 Verifying the model

The aim of the verification process is that similar properties, indeed patterns, expressing user interface characteristics can be checked of each candidate device. These formal properties provide a benchmark against which each candidate design can be explored. As a result, potentially unforeseen design consequences can be discovered that could not be found simply by reading the manual or by experimenting with the device. The circumstances in which properties fail are assessed with the help of human factors or domain expertise. Failure acts as a trigger for the consideration of a human interface characteristic that would otherwise lie hidden. Properties checked of each candidate are of the following types.

- Checking that the process represented in the innermost pump layer is visible through the device interface (mirroring the process in the interface).
- Checking that modes can be determined unambiguously from the interface (mode clarity).
- Checking that actions provide appropriate feedback, for example when they change mode or change the values of pump attributes.
- Ensuring consistency of use of the display, or of action (consistency of the interface).
- Checking ease of recovery from an action.
- Ensuring that activities described in the outer layer are supported (supporting activities).

Many of these properties are familiar and can be seen as interpretations of principles described in [18] or [32]. The specific details of the properties in these categories will differ depending on the device's interface as is described in the middle layer of the model. The aim is to instantiate standard templates as far as possible. In the case of the devices modelled here, the properties that were checked of the pump and interface layers were checked first before considering the devices constrained by activity assumptions. Properties are presented for analysis using CTL (see [16] for an introduction to model checking and CTL). The properties are analysed in the IVY tool by translating the MAL model into SMV and using the symbolic model checker NuSMV [15] to check the CTL properties. It was necessary to restrict the state space by reducing the ranges of vtbi, infusion rate and menu lengths. Checking the two layer model is a relatively slow process, for example the set of properties described in the paper for the Alaris required approximately four hours of elapsed time of a laptop using a Dual 1.86 GHz Intel Core 2 Duo with 3893 MB of RAM.

Each of the properties is now described in turn.

5.1 Mirroring the process in the interface

The analysis measures the extent to which the infusion pump process is visible in the device. This visibility relates to basic pump variables: infusion rate, vtbi, time to infuse. It is also concerned with whether or not the pump is infusing and whether or not, while infusing, it has exhausted vtbi and is in KVO mode. The middle layer (interface) model describes the interface to these attributes by recording whether or not they are displayed. Visibility is modelled using a set of Boolean attributes. Hence *vtbi* is visible in the Alaris pump if $middisp[dvtbi]$ is true and in the case of the B.Braun device if $disp[dvtbi]$ is true. The pump device has two modes. The first, infusion status, describes whether or not the pump is infusing. The second distinguishes between normal infusion and KVO infusion. The two modes are signified by two attributes in the device model. *infusionstatus* takes three values namely *infuse*, *hold* and *blank*. *blank* is a value that signifies that the pump device is off. *kvoflag* is a boolean that determines whether or not the device is in KVO mode.

In the case of the Alaris pump the most significant predictor of what the device is doing is the top line, see Figure 3. The B.Braun displays mode related information in different places. This mode display information is represented in the B.Braun model by using the state attribute *displaymode*. Whether infusion status is represented unambiguously can be assessed by checking display properties relating to the status. In the case of Alaris the question leads to a fairly complicated answer. The Alaris displays the same top line information while both infusing and holding in a number of circumstances because it is possible to adjust aspects of the process while the device is infusing. The B.Braun is less complex from this perspective. The process of checking standard properties is itself a discovery process. Often standard properties of the device fail because they are only partially true. It becomes necessary to explore counter-examples and to add further constraints until a true property of appropriate complexity is determined.

$$AG((device.infusionstatus! = blank) \rightarrow (topline\ in\ \{infusing, dispkvo\} \leftrightarrow device.infusionstatus = infuse)) \quad (2)$$

Property 2 shows a stage in the development of the relevant property for the Alaris. Properties must exclude the possibility of the device being switched off (expressed as the infusion status not being blank). The property indicates that infusing status is indicated by a top line that either displays "infusing" or "KVO". This property turns out to be false because the device can continue to infuse when clearing volume, changing vtbi, when indicating that the device is locked, and when selecting certain options using the *query* button and when showing the information associated with an option. These top lines can appear in both infusing and holding states. In the case of the infusion status being "hold" Property 3 indicates a more refined stage in the development of the property for the Alaris.

$$AG((device.infusionstatus\ != blank)\ \&\ !(topline\ in\ \{locked, volume, options, dispinfo, dispvtbi\} \rightarrow (topline\ in\ \{holding, setvtbi, attention, vtbitime\} \leftrightarrow device.infusionstatus = hold))) \quad (3)$$

The question that equations 2 and 3 raise is whether the multiplicity of exceptions is a good or a bad thing. It is at this stage that further input from domain or human factors experts is required. Potential ambiguities may not be an issue because other features of the Alaris interface may make the status of the device clear to the user. Indeed aspects of the design, wholly unrelated to this interface, may be critical, for example the sound of the pump operating. The analysis has raised these issues in a systematic way that can lead to the appropriate discussion. Relevant situations would be explored by taking traces that indicate where properties fail as a starting point. Domain and human factors experts can use the sequences to inspire consideration of scenarios to explore whether the failure of the property in this situation will be a problem in practice.

The B.Braun pump is designed differently. There are a number of ways in which mode is indicated. These are specified in the model in the single attribute *displaymode*. These different modes can be indicated, for example, by the size of the label referring to the data entry display. A separate discussion with human factors and domain experts would lead to consideration of whether these indications are sufficiently salient to distinguish the elements of *displaymode*. It is assumed by the model that this feature (whatever it is) is salient for the user. In the case of infusing, the display shows arrows that move dynamically as the pump progresses. The simple standard property that began the Alaris investigation of pump status turns out to be true immediately for the B.Braun.

$$AG((device.infusionstatus\ != blank) \rightarrow (displaymode = dispinfusing) \leftrightarrow device.infusionstatus = infuse)) \quad (4)$$

There are many more display modes in the case of the B.Braun that indicate an infusion status of holding.

$$AG((device.infusionstatus\ != blank) \rightarrow (displaymode\ in\ \{disprate, dispvtbi, disptime, mainmenu, dispalarm, \quad (5)$$

$$dispalarmvol, optionsmenu, statusmenu, dispblank\})$$
$$\leftrightarrow device.infusionstatus = hold)$$

Property 5 unambiguously defines the holding behaviour of the device.

5.2 Checking ambiguity of modes

There are a number of properties that explore the ambiguity or otherwise of interface modes.

1. Does the display unambiguously determine the mode of the device? Key attributes here are *topline* for the Alaris and *displaymode* for the B.Braun
2. Are the mode relevant pump variables visible in the relevant mode?

Exploring the first question in the two devices produces interesting results. For example, the modification of vtbi in the Alaris can be achieved in three modes. It can be changed manually via the data entry keys when "vtbi" appears as the top line. vtbi can also be entered along with time rather than with infusion rate (a prescription may require that a particular volume be infused over a period of time rather than at a particular infusion rate) by selecting an appropriate options menu entry. It is also possible to access a menu of presets (bag mode) either from normal vtbi entry mode when "vtbi" appears as the topline or from vtbi entry over time when "vtb/time" appears in the top line. The simplest mechanism for entering vtbi assuming the specification of infusion rate is also accessible when *infusing* including the option of selecting an infusion bag using the bag menu, but it is not possible to enter vtbi over time. These modes are distinguished by values of the attribute *entrymode*: *vtmode*, *bagmode* (when with infusion rate) and *vttmode* and *tbagmode* (when with time). It might be expected that either the device clearly distinguishes the four modes through the top line or uses the top line to make it clear that all modes are about entering vtbi. In fact neither situation is the case:

$$AG(entrymode\ in\ \{vtmode, bagmode, tbagmode\} \leftrightarrow topline = dispvtbi) \quad (6)$$

is true. But this does not include *vttmode* which is the mode of entry of vtbi when entering vtbi over time.

$$AG(entrymode = vttmode \leftrightarrow topline = vtbitime) \quad (7)$$

is false because the time entry mode *ttmode* also occurs when $topline = vtbitime$ (the distinction between the modes is indicated by the position of an arrow). The B.Braun does not exhibit such ambiguities. The B.Braun model includes for example a data entry mode which is true when the display mode indicates entry of vtbi, time and infusion rate.

$$AG(displaymode\ in\ \{dispvtbi, disprate, disptime\} \leftrightarrow entrymode = dataentry) \quad (8)$$

Similar properties are true of *scalemode* and *alarmmode*.

$$AG(displaymode = dispalarmvol \leftrightarrow entrymode = scalemode) \quad (9)$$

$$AG(displaymode = dispalarm \leftrightarrow entrymode = alarmmode) \quad (10)$$

Property 10 is false because there are other situations that will generate alarm mode. The second feature of the two devices that is important in considering modes is whether the pump variables that are being modified are visible in the relevant mode. For example it should be expected that the infusion rate is visible when in a mode in which infusion rate is being entered. In the case of the Alaris the infusion rate can be entered either in *infusemode* or *rmode*:

$$AG(entrymode\ in\ \{rmode, infusemode\} \rightarrow middisp[drate]) \quad (11)$$

This property of the model is true and similar properties relate to other modes, for example:

$$AG(entrymode = vtmode \rightarrow middisp[dvtbi]) \quad (12)$$

$$AG(entrymode\ in\ \{bagmode, tbagmode\} \rightarrow middisp[dvtbi]) \quad (13)$$

While Properties 11 and 12 are true of the device, Property 13 is not. The current value of vtbi is not visible in *bagmode* or *tbagmode*. Whether this is a problem depends on whether or not the current value of vtbi is relevant when choosing a new bag to be infused. This issue would raise a discussion about the work context. It is likely for example that a bag will be selected only at the start of a new infusion process rather than as an addition to an existing infusion process.

In the case of the B.Braun, device pump variables are visible in four situations.

1. They are visible when the pump is infusing then infusion rate, vtbi, time and volume infused are visible.
2. They are visible when the main menu is being shown and infusion rate, vtbi and time are the menu options that are visible. In this case the three variables are visible.
3. They are visible in the relevant data entry mode.
4. They are visible when alarming if the situation before the alarm was one of the above.

The situations in which vtbi is visible are identified through the following property.

$$AG(((device.infusionstatus \; != \; blank) \; \& \quad (displaymode \; != \; dispalarm)) \rightarrow (disp[dvtbi] \leftrightarrow ((displaymode = mainmenu \; \& \; menucursor < dvol) \mid (displaymode \; in \; \{dispvtbi, dispinfusing\}))))) \tag{14}$$

The visibility of the infusion rate is identified through the following analogous property.

$$AG(!(displaymode \; in \; \{mainmenu, dispinfusing, dispalarm\}) \rightarrow (displaymode = disprate \; \& \; entrymode = dataentry \leftrightarrow disp[drate])) \tag{15}$$

The analysis of mode in the two devices has revealed an Alaris device that has a considerably more complex mode structure than the B.Braun. It is possible in the case of the Alaris to perform a number of activities while the pump is infusing which is not possible with the B.Braun. The Alaris supports four different "vtbi entry" modes when the device is holding and two of them can be used when the device is infusing. This additional complexity is not necessarily a bad thing. Rather the analysis raises issues that can be further explored from a domain or human factors perspective. It is possible for example that this complexity improves the efficiency if supported by training and a natural and supportive work structure.

5.3 Checking feedback of actions

A number of issues may be explored that relate to checking feedback of actions. Here the feedback template [10] can be applied. These issues include two questions.

1. Will a function key change mode, and will this change of mode be visible?
2. Does pressing a data entry key always change the pump attribute relevant to the mode, and is this change visible?

These issues are explored through sets of properties. The first set relates to the visible expression of mode while the second relates to the internal representation. Hence in the case of the Alaris these properties are investigated via the following templates. The visibility properties are:

$$AG(topline = \texttt{value} \rightarrow AX(\texttt{function} \; \rightarrow \; topline \; != \; \texttt{value})) \tag{16}$$

The properties that check the change of mode are as follows.

$$AG(entrymode = \texttt{value} \; \rightarrow AX(\texttt{function} \rightarrow entrymode = \texttt{value})) \tag{17}$$

Checking this set of properties indicates that:

1. pressing *key3* marked as "cancel" leaves the device in infuse mode when the top line shows "vtbi done".
2. *tick* never changes the *entrymode*.

Otherwise the keys will always change the function. The second issue can be explored through a set of properties, see [11], that check the attributes that are changed by number entry keys in particular modes.

$$AG(entrymode = \texttt{IVAL1} \; \& \; \texttt{modeattribute} = \texttt{IVAL2} \rightarrow \; AX(\texttt{action} \rightarrow \; entrymode = \texttt{IVAL1} \; \& \; \texttt{modeattribute} \; != \; \texttt{IVAL2})) \tag{18}$$

`IVAL1` here is a meta-variable that allows IVY to instantiate and to verify the property with all possible values for *entrymode* and `IVAL2` to all possible values of the mode attribute. Hence the following instance of the set is true.

$$AG(entrymode = vtmode \; \& \; device.vtbi = 3 \rightarrow \; AX(sup \rightarrow \; entrymode = vtmode \; \& \; device.vtbi \; != \; 3)) \tag{19}$$

In the case of the B.Braun the relevant properties are:

$$AG(displaymode = \texttt{value} \rightarrow AX(\texttt{function} \; \rightarrow \; displaymode \; != \; \texttt{value})) \tag{20}$$

and

$$AG(entrymode = \texttt{value} \; \rightarrow AX(\texttt{function} \rightarrow entrymode = \texttt{value})) \tag{21}$$

Checking these properties has the expected effect.

5.4 Checking consistency of action

As discussed in previous sections the two devices support function keys that change modes. In the case of Alaris the function keys are soft keys, they are labeled (*fndisp1*, *fndisp2*, *fndisp3*) in the model. The B.Braun keys are not soft keys but often how the key is to be interpreted is indicated in the display. For example the infusion data entry mode labels "OK" as "confirm". The main function of the B.Braun display in relation to function keys is to indicate their availability. Two types of consistency are explored here. The differences between the devices means that a number of properties are only relevant to one of the devices. Hence in the case of the Alaris the obvious questions are whether the same soft function is always associated with the same key and whether the same soft keys are always associated with a particular piece of information on the

top line. To illustrate how the model can be analysed with respect to these questions, consider *key3*. A first question would be whether "quit" is always associated with *key3* whenever it is used.

$$AG(fndisp3 \; != \; fnull \rightarrow fndisp3 = fquit) \qquad (22)$$

This property turns out to be false. There are a number of situations where *key3* is used for other purposes. In bags mode function *key3* shows "back" rather than "quit". When the top line shows "attention" or "vtbi done" or "set vtbi" then *key3* is marked as "ok" and used to exit the dialogue box. Exploring all these cases leads to the following true property.

$$\begin{aligned} AG((&fndisp3 \; != \; fnull) \; \& \qquad (23)\\ &(!(entrymode \; in \; \{bagmode, tbagmode\}) \; \&\\ &!(topline \; in \; \{attention, vtbidone, setvtbi\}))\\ &\rightarrow fndisp3 = fquit) \end{aligned}$$

It can be shown however that if the soft function is "quit" then it always appears as *key3*. This can be demonstrated by checking:

$$\begin{aligned} AG((&fndisp1 = fquit \mid fndisp2 = fquit \mid \qquad (24)\\ &fndisp3 = fquit) \; \rightarrow \; fndisp3 = fquit) \end{aligned}$$

Furthermore general configurations of function keys can be asserted:

$$\begin{aligned} AG(&topline = volume \rightarrow \qquad (25)\\ &(fndisp1 = fnull \; \& \; fndisp2 = fclear\\ &\& \; fndisp3 = fquit)) \end{aligned}$$

It can also be shown that in most circumstances the same top line is always associated with the same key configuration. Property 26 is a property that fails.

$$\begin{aligned} AG(&topline = dispvtbi \rightarrow \qquad (26)\\ &(fndisp1 = fok \; \& \; fndisp2 = fbags \; \&\\ &fndisp3 = fquit)) \end{aligned}$$

In bags mode, top line displays "vtbi" but the function keys show "ok", "null" and "back" respectively. The Alaris and B.Braun function keys can be compared more directly using properties similar to the feedback template supported by the IVY tool [10]. Several applications are relevant here in the case of the Alaris.

$$\begin{aligned} AG((&fndisp1 = fok \; \& \; topline \; != \; options) \; \rightarrow\\ &AX(key1 \; \rightarrow \; topline = holding)) \qquad (27) \end{aligned}$$

$$\begin{aligned} AG(&fndisp3 = fquit \; \rightarrow \qquad (28)\\ &AX(key3 \; \rightarrow \; topline = holding)) \end{aligned}$$

$$\begin{aligned} AG((&topline = volume \& infusionstatus = hold)\\ &\rightarrow \; AX(key2 \rightarrow volumeinfused = 0)) \qquad (29) \end{aligned}$$

Properties 27 - 29 indicate a sample of these consistency properties. They indicate, for example, that in the case of Property 27 "ok" always returns the device to a top line of "holding" except when the top line shows "options" (a special mode for changing settings and entering vtbi over time). Property 28 confirms that "quit" always returns to the "hold" state and Property 29 indicates that when top line is volume and *key2* is pressed the volume infused is initialized. The B.Braun has similar properties which are more direct instantiations of the feedback template.

$$\begin{aligned} AG(&device.infusionstatus = hold \; \& \qquad (30)\\ &displaymode \; != \; mainmenu \; \rightarrow\\ &AX(ok \; \rightarrow \; displaymode = mainmenu)) \end{aligned}$$

The effect of action is also consistent in the sense that the temporary value being entered is always used to update the attribute relevant to the mode when the *key* is used.

$$\begin{aligned} AG((&displaymode = disptime) \; \& \qquad (31)\\ &(dispinftime = \texttt{IVAL1}) \; \&\\ &(device.time \; != \; \texttt{IVAL1}) \; \rightarrow\\ &AX(ok \; \rightarrow \; device.time \; = \; \texttt{IVAL1})) \end{aligned}$$

`IVAL1` here allows IVY to instantiate and to verify the property with all possible values for *device.time* and *disptime*. Property 31 holds for all values of `IVAL1`. The multiple uses of function keys in the case of the Alaris adds to the complexity of its interface. It is important however to see how these two devices behave in relation to the activities that these two designs are intended to support. This issue is discussed in a little more detail when discussing activities in the next section. Data entry keys have quite different characteristics in the two devices. The Alaris chevron keys increment or decrement the displayed value either by small steps (sup, sdown) or by big steps (fup, fdown). The B.Braun keys either move the cursor (left, right) or increment or decrement the digit indicated by the cursor (up, down). A detailed analysis of number entry for these devices can be found in [34]. The model described has also been translated into an equivalent model in which the actual number domains are used and subjected to theorem proving as will be discussed in a future paper. The properties to prove are examples of the guarded consistency pattern [10] which demonstrate that whatever the data entry mode, the keys have a similar effect on the relevant pump variable. Hence the standard form of the property is that whatever the mode, the chevron key will have the relevant effect on the corresponding mode attribute.

$$\begin{aligned} AG((&entrymode = mode \; \& \; (\texttt{modeattribute})) \; \rightarrow\\ &AX(\texttt{chevron} \; \rightarrow \; \texttt{effect}(\texttt{modeattribute}))) \qquad (32) \end{aligned}$$

For example in *rmode* when the infusion rate lock is off the single chevron up button increments the infusion rate by 1.

$$AG((!rlock\ \&\ entrymode = rmode\ \&\ (infusionrate = \texttt{IVAL1})) \rightarrow AX(sup \rightarrow infusionrate = \texttt{IVAL1} + 1)) \quad (33)$$

The corresponding property for the B.Braun is more complicated because the effect is defined in terms of the different elements of the numeral that are significant depending on the position of the cursor. In the B.Braun model the cursor is associated with attribute entry and *dispvalue* is the attribute that is manipulated in data entry mode before updating the relevant attributes.

$$AG((entrymode = dataentry)\ \&\ (dispvalue = \texttt{IVAL}) \rightarrow AX(down \rightarrow ((entry = 3 \rightarrow dispvalue = \texttt{IVAL} - 1)\ \& (entry = 2 \rightarrow dispvalue = \texttt{IVAL} - 2)\ \& (entry = 1 \rightarrow dispvalue = \texttt{IVAL} - 4)\ \& (entry = 0 \rightarrow dispvalue = \texttt{IVAL} - 8)))) \quad (34)$$

In practice the property needs adjusting to deal with values that are too large or too small as constrained by the invariant in axiom 1 of the pump layer (Section 4.2), these cases must exclude the possibility of underflow or overflow. In addition to the standard Property 34, of the same form as the template Property 30, a further property is required to ensure that when data entry mode is exited (using *ok*) then the relevant pump attribute depending on displaymode is updated appropriately. The following property shows how this works with time entry. *displaymode* = *disptime* only when *dataentry* is true and therefore the property has been simplified a little.

$$AG((displaymode = disptime)\ \&\ (dispinftime = \texttt{IVAL})\ \&\ (device.time\ !=\ IVAL) \rightarrow AX(ok \rightarrow device.time = \texttt{IVAL})) \quad (35)$$

5.5 Checking ease of recovery

The final stage of analysis of the two devices concerns the ease with which the two devices can recover from a wrong action. There are a variety of properties to be explored in this context. One that can be used to illustrate the analysis is to demonstrate that the data entry keys always have inverses. This property has a standard template [10].

$$AG(attribute = value \rightarrow AX(\texttt{action1} \rightarrow EX(\texttt{action2})\ \& AX(\texttt{action2} \rightarrow (\texttt{attribute} = \texttt{value})))) \quad (36)$$

In the case of *sup* in the Alaris pump and *rmode* when the infusion rate is not locked the following property holds for all values of `IVAL1` except the limits.

$$AG((infusionrate = \texttt{IVAL1}\ \&\ entrymode = rmode\ \&\ !rlock) \rightarrow AX(sup \rightarrow (EX(sdown)\ \&\ AX(sdown \rightarrow infusionrate = \texttt{IVAL1})))) \quad (37)$$

The recovery property in the case of B.Braun is complicated because of "carry". When incrementing the digit in the leftmost position of the numeral, the model specifies that "1000" goes to "1111". This is the largest allowable number given the constraints of the domains that have been reduced to permit analysis by model checking. When the left most digit is decremented then this value returns to "0111". When numbers are not subject to this overflow condition then down acts as an inverse.

$$AG(entrymode = dataentry\ \&\ dispvalue = \texttt{IVAL1} \rightarrow AX(up \rightarrow ((EX(down)\ \&\ AX(down \rightarrow (entrymode = dataentry\ \&\ dispvalue = \texttt{IVAL1})))))) \quad (38)$$

Perhaps more relevant is the ability of the devices to allow a sequence of actions to reach a different number from whatever the current value. This is a property that can also be proved.

5.6 Checking support for activities

The support of activities is explored by proving that specific goals related to the infusion pump can be achieved. The activity associated with prescription values of vtbi of 6 ($mvolume = 6$) and time infused of 3 ($mtime = 3$) can be checked in the model with all three layers. Counter-examples that will result from failure of Property 39 will be subject to the constraints imposed by the third layer of the model.

$$AG(device.volumeinfused\ !=\ mvolume) \quad (39)$$

Alaris activities

The trace that specifies a counter-example (see Figure 4) indicates:

1. Use of the query mode, first to lock the infusion rate (to prevent further modification) then to choose the option that is used to enter vtbi over time (steps 3-9).
2. The enter vtbi activity in which double and single chevron keys are used to reach the prescribed value of vtbi (steps 10-14).

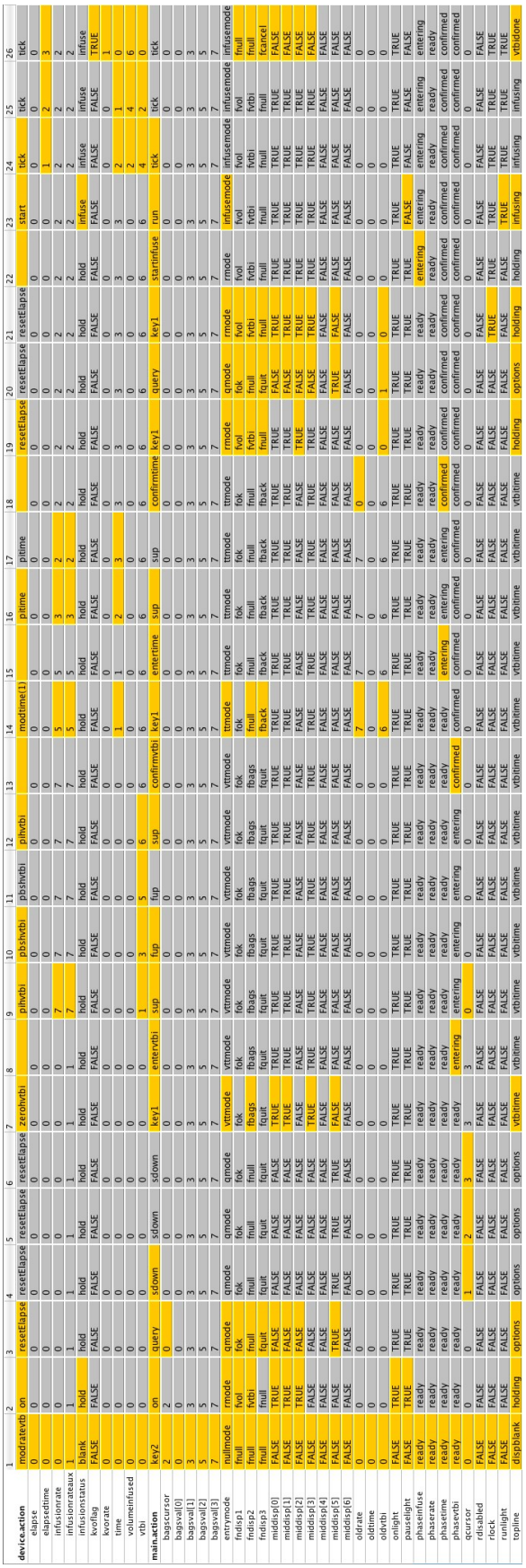

Fig. 4 The Alaris activity counter-example

3. Confirmation that vtbi has been entered (step 15).
4. The enter time activity using the single chevron up button (steps 17-19).
5. Confirmation of the time (step 20)
6. Commencement of infusing (step 24)
7. Infusion until the vtbi is exhausted (step 25 onwards).

This sequence can be contrasted with that of the B.Braun (see Figure 5):

1. The main menu item is chosen and the enter vtbi activity is carried out (steps 5-12). vtbi is entered via a sequence of *up* buttons and this requires further exploration.
2. The vtbi is confirmed (step 13).
3. The time choice is selected and the enter time activity commenced (steps 16-20).
4. Time is confirmed (step 21).
5. Start infusing at step 22.

It can be seen that the model checker has generated plausible sequence of actions in both cases. This sequence could be further analysed in a more qualitative way using a clinician or human factors specialist to explore the implications of these sequences.

6 Discussion and Conclusions

This analysis of the two infusion pumps has provided solid evidence of areas of concern where redesign would reduce complexity and increase the reliability of the two designs. The properties used in the analysis are largely based on standard patterns either generated by the IVY tool or easily generated from the requirements of the device and situation. Analogous properties can be systematically applied to other candidate pumps. The procedure is systematic providing a strong basis for comparison.

Specifically, the analysis has indicated issues associated with the relation of the display to the underlying pump process and the mode structure of the particular device. For example, the analysis indicates that there are:

- possible confusions relating to the use of the display of vtbi in the top line in the case of the Alaris;
- ambiguities about whether the pump is infusing or not using the top line of the display as a guide in the case of the Alaris;
- inconsistent use of function keys in the case of the Alaris;
- possible confusions combining left, right, up and down in the case of the B.Braun.

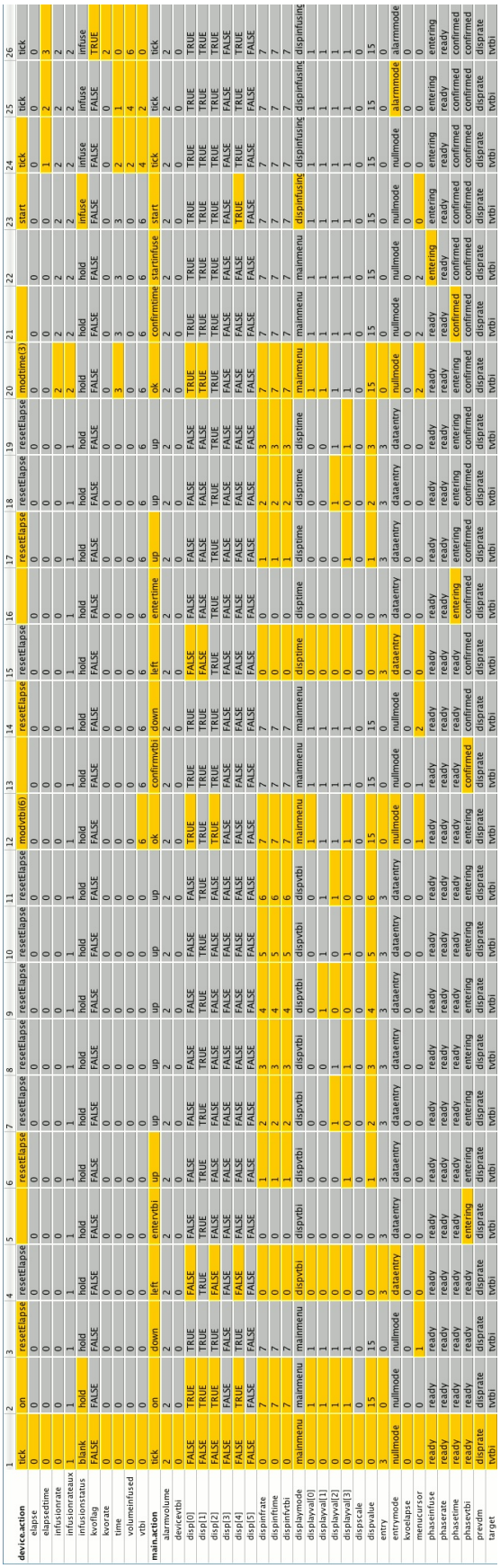

Fig. 5 The B.Braun activity counter-example

The analysis says nothing about the significance of these issues. The method is to be used as part of a process including the active participation of human factors and domain specialists. In the context of use these discrepancies in the device may not be issues at all.

An important aspect of the method is that it is systematic and that it has the potential to be reused for every candidate infusion pump. The pump layer of the specification can be reused for every candidate pump. The outer activity layer can also be partially reused, guiding the analyst to create the appropriate constraints on the device model. The interface layer will be created afresh for each new device. In practice this part of the model continues to be substantial (perhaps a week's work for the first author). Note also that the approach is not restricted to infusion pumps. It can be used for any device that controls a process over time, and indeed for any interactive computing system if we consider the inner ("pump") layer as the functional (or business logic) layer of the system. In fact, earlier forms of the approach have been used in a number of different devices, as discussed in Section 2.

One recurring concern in formal verification relates to model veracity. In the current case the goal has been to reason about the devices' design. That is, the models capture the design of the systems. Hence, their veracity is judged against the understanding of how the system is supposed to work. This was achieved by systematic exploration of the devices interfaces in order to build an understanding of their operation.

Further work is required to develop methods of reusing models that have a similar structure or drive particular mode structures. It can be argued that there is still some distance to go before procurers or developers will find the techniques described here cost-effective, however the cost of using an infusion device extends beyond making a good deal on price and ongoing maintenance. The work described here is part of the ongoing analysis of a range of infusion devices with the aim of reducing these costs. Part of this process should include a comparison with empirical techniques and with usability evaluation methods in terms of the range of problems that can be uncovered by these different methods.

Acknowledgements This project was partly funded by the CHI+MED project: Multidisciplinary Computer Human Interaction Research for the design and safe use of interactive medical devices (UK EPSRC Grant EP/G059063/1). Patrick Oladimeji of Swansea University provided help with the Alaris pump and Chris Vincent of UCL provided access to the B.Braun simulation. We are grateful to reviewers for helpful comments.

References

1. T. Amnell, G. Behrmann, J. Bengtsson, P.R. D'Argenio, A. David, A. Fehnker, T. Hune, B. Jeannet, K.G. Larsen, M.O. Möller, P. Pettersson, C. Weise, and W. Yi. UPPAAL - Now, Next, and Future. In F. Cassez, C. Jard, B. Rozoy, and M. Ryan, editors, *Modelling and Verification of Parallel Processes*, number 2067 in Lecture Notes in Computer Science Tutorial, pages 100–125. Springer–Verlag, 2001.
2. D. Arney, B. Kim, R. Jetley, P. Jones, I. Lee, A. Ray, O. Sokolsky, and Y. Zhang. Safety requirements for the generic patient controlled analgesia pump.
3. B. Braun Melsungen AG. B.Braun Infusomat Space User Manual. Technical report, B. Braun Melsungen AG, 34209 Melsungen, Germany, 2007.
4. L-A. Bligard and A-L. Osvalder. An analytical approach for predicting and identifying use error and usability problem. In A. Holzinger, editor, *Proceedings of the 3rd Human-computer interaction and usability engineering of the Austrian computer society conference on HCI and usability for medicine and health care*, number 4799 in Springer Lecture Notes in Computer Science, pages 427–440. Springer-Verlag, 2007.
5. M. L. Bolton and E. J. Bass. Formally verifying human-automation interaction as part of a system model: limitations and tradeoffs. *Innovations in System and Software Engineering*, 6(3):219–231, 2010.
6. M. L. Bolton, E. J. Bass, and R. I. Sininiceanu. Generating phenotypical erroneous human behavior to evaluate human-automation interaction using model checking. *International Journal of Human-Computer Studies*, 70:888–906, 2012.
7. M. L. Bolton, E.J. Bass, and R.I. Siminiceanu. Using formal verification to evaluate human-automation interaction, a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, in press.
8. J. C. Campos. Minho HCI repository. `http://hcispecs.di.uminho.pt`, December 2012.
9. J. C. Campos and M. D. Harrison. Model checking interactor specifications. *Automated Software Engineering*, 8:275–310, 2001.
10. J. C. Campos and M. D. Harrison. Systematic analysis of control panel interfaces using formal tools. In N. Graham and P. Palanque, editors, *Interactive systems: Design, Specification and Verification, DSVIS '08*, number 5136 in Springer Lecture Notes in Computer Science, pages 72–85. Springer-Verlag, 2008.
11. J. C. Campos and M. D. Harrison. Interaction engineering using the IVY tool. In G. Calvary, T.C.N. Graham, and P. Gray, editors, *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 35–44. ACM Press, 2009.
12. J. C. Campos and M.D. Harrison. Modelling and analysing the interactive behaviour of an infusion pump. *Electronic Communications of the EASST*, 5, 2011.
13. José C. Campos and Michael D. Harrison. Formally Verifying Interactive Systems: A Review. In M.D. Harrison and J.C. Torres, editors, *Proceedings on the 4th Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSVIS)*, pages 119–134. Springer-Verlag, 1997.
14. Cardinal Health Inc. Alaris GP volumetric pump: directions for use. Technical report, Cardinal Health, 1180 Rolle, Switzerland, 2006.
15. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An Open Source Tool for Symbolic Model Checking. In K. G. Larsen and E. Brinksma, editors, *Computer-Aided Verification (CAV '02)*, volume 2404 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.
16. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
17. L. de Moura. SAL: Tutorial. Technical report, SRI International, Computer Science Laboratory, 333 Ravenswood Avenue, Menlo Park, CA 94025, 2004.
18. A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction*. Prentice Hall, 1993.
19. A. J. Dix. *Formal Methods for Interactive Systems*. Academic Press, 1991.
20. G. Doherty, J. C. Campos, and M. D. Harrison. Resources for situated actions. In N. Graham and P. Palanque, editors, *Interactive systems: Design, Specification and Verification, DSVIS '08*, volume 5136 of *Springer Lecture Notes in Computer Science*, pages 194–207. Springer-Verlag, 2008.
21. D. J. Duke and M. D. Harrison. Abstract interaction objects. *Computer Graphics Forum*, 12(3):25–36, 1993.
22. J. Fiadeiro, T. Maibaum, J. de Bakker, W. de Roever, and G. Rozenberg. Describing, structuring and implementing objects. In *Foundations of Object-Oriented Languages*, number 489 in Springer Lecture Notes in Computer Science, pages 274–310. Springer-Verlag, 1991.
23. R. E. Fields. *Analysis of erroneous actions in the design of critical systems*. PhD thesis, Department of Computer Science, University of York, Heslington, York, YO10 5DD, 2001.
24. US Food and Drug Administration. Infusion pump improvement initiative. Technical report, Center for Devices and Radiological Health, April 2010.
25. M. Heymann and A. Degani. Formal analysis and automatic generation of user interfaces: Approach, methodology, and an algorithm. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 49(2):311–330, 2007.
26. P. N. Johnson-Laird. *Mental Models*. Harvard University Press, 1983.
27. B. Kim, A. Ayoub, O. Sokolsky, I. Lee, P. Jones, Y. Zhang, and R. Jetley. Safety-assured development of the GPCA infusion pump software. In *Proceedings of the ninth ACM international conference on Embedded software*, EMSOFT '11, pages 155–164, New York, NY, USA, 2011. ACM.
28. B. Kirwan and L. Ainsworth. *A Guide to Task Analysis*. Taylor and Francis, 1992.
29. R. Lane, N. A. Stanton, and D. Harrison. Applying hierarchical task analysis to medication administration errors. *Applied Ergonomics*, 37:669–679, 2006.
30. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, Oct 1997.
31. 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.
32. J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In J. Chew and J. Whiteside, editors, *ACM CHI Proceedings CHI '90: Empowering People*, pages 249–256, 1990.
33. P. Oladimeji. Alaris simulation. `http://cs.swan.ac.uk/~cspo/simulations`, December 2012.
34. P. Oladimeji, H. Thimbleby, and A. Cox. Number entry and their effects on error detection. In P. Campos et al.,

editors, *Interact 2011*, number 6949 in Springer Lecture Notes in Computer Science, pages 178–185. Springer-Verlag, 2011.
35. F. Paternò and G. Faconti. On the Use of LOTOS to Describe Graphical Interaction. In A. Monk, D. Diaper, and M. D. Harrison, editors, *People and Computers VII: HCI '92 Conference*, pages 155–174. BCS HCI Specialist Group, Cambridge University Press, 1992.
36. R. Ruksenas, J. Back, P. Curzon, and A. Blandford. Verification-guided modelling of salience and cognitive load. *Formal Aspects of Computing*, 21:541–569, 2009.
37. M. Ryan, J. Fiadeiro, and T. Maibaum. Sharing actions and attributes in modal action logic. In *Theoretical Aspects of Computer Software*, volume 526 of *Springer Lecture Notes in Computer Science*, pages 569–593. Springer-Verlag, 1991.
38. H. Thimbleby and P. Cairns. Reducing number entry errors: Solving a widespread, serious problem. *Journal Royal Society Interface*, 7(51):1429–1439, 2010.
39. H. W. Thimbleby and A. Gimblett. Dependable keyed data entry for interactive systems. *ECEASST*, 45, 2011.
40. H.W. Thimbleby. Interaction walkthrough: evaluation of safety critical interactive systems. In G. Doherty and A. Blandford, editors, *Interactive Systems: Design, Specification and Verification*, number 4323 in Springer Lecture Notes in Computer Science, pages 52–66. Springer-Verlag, 2007.
41. H.W. Thimbleby. *Press on: principles of interaction programming*. MIT Press, 2007.
42. J. Zhang, T. R. Johnson, V. L. Patel, D. L. Paige, and T. Kuboseb. Using usability heuristics to evaluate patient safety of medical devices. *Journal of Biomedical Informatics*, 36:23–30, 2003.