Templates as heuristics for proving properties of medical devices

José Creissac Campos
Dep. Informática /
Universidade do Minho &
HASLab / INESC TEC
Braga, Portugal
jose.campos@di.uminho.pt

Paul Curzon and Paolo Masci
EECS, Queen Mary University of London
Mile End, London E1 4NS, UK
p.curzon@qmul.ac.uk,
p.m.masci@qmul.ac.uk

Michael Harrison
School of Computing Science,
Newcastle University,
Newcastle-upon-Tyne, UK
Universidade do Minho &
HASLab/INESC TEC, Queen
Mary University London
michael.harrison@ncl.ac.uk

ABSTRACT

This paper briefly describes how property templates have been used to analyse and explore the interactive behaviour of a specific medical device (an IV infusion pump). It is proposed that interactive devices that satisfy properties based on the templates are easier and safer to use. The property templates act as heuristics for the development of suitable properties tailored to the details of the particular device. A mathematically based approach is used to prove that a specification of the device satisfies the properties.

Author Keywords
formal methods, interactive systems, usability heuristics

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

As tools for the mathematically based specification and analysis of software systems become more robust [8], the potential for their use in the development and safety analysis of critical systems is increasing. This short paper illustrates how a theorem proving approach has been used to analyse safety related properties of an interactive device. The broader focus of our analysis has been intravenous infusion and haemodialysis. These devices are susceptible to (sometimes fatal) failure through use error. The US Food and Drugs Administration (FDA) are aware of these safety concerns and have been exploring the role of safety requirements (some of which are user-centred) to facilitate safety analysis [7, 1]. These safety requirements are designed to mitigate potential consequences of hazards, reducing risks in the design of the devices. We propose a set of generic property templates that can be used to generate appropriate safety requirements for interactive devices. The templates are illustrated using the design of an existing widely used IV infusion pump.

THE MODEL

The analysis begins with a specification of interactive aspects of the behaviour of the device. This has been done using a particular theorem proving system (PVS [6]). The specification of the device under analysis is described in terms of states and actions over states. To capture the shape of the specification we take a more abstract and generic version of it. States are described by attributes, for example basic attributes such as pump variables or perceivable attributes such as the information displayed in the top line of the display. Actions are defined that transform states. Actions can change pump variables, they can change the *mode* of the device, they can change the display. Actions can be user actions or autonomous actions such as describe the process of the pump.

Use errors arise because the effect of an action is not visible, or more importantly *not noticed*. There are several reasons why this might happen — the behaviour of the action is unexpected, either because the device is in a different mode (it is changing the volume to be infused rather than the infusion rate), or because an action behaves unexpectedly in a given circumstance, for example a key that reverses an effect (a $down$ key when entering a number, reverses an up key in *almost* all cases but not all).

Some elements of the state of the interactive system are perceivable (for example, visible or audible). Autonomous and user actions transform the state [4]. Furthermore, not all actions are permitted all of the time and the behaviour of actions can depend on state attributes called *modes*. Modes appear when there is a need to *multiplex* several possible user commands into a single physical interface element. They may lead to confusion for users about what the effect of an action will be.

A basic model used to represent interactive systems can therefore be described as a set of actions $a : A = S \longmapsto S$ where S is a set of states. A state is itself a set of attributes. Actions are partial functions. They are made total by including a value “undefined” ($\perp$). A function *per* takes an action and determines whether it is defined for a value in its domain $per : A \rightarrow (S \rightarrow T)$ such that $per(a)(s) = true$ if $a(s) \neq \perp$.

Some elements of the state are part of the interface. Attributes of the state that are perceivable are defined by $p : S \rightarrow P$ (where P is a set of perceivable attributes) and a relation $VIS = S \times P$ determines whether there is a perceivable attribute related to an attribute in the state S. The perceivable attributes will be identified and selected in formulating the property.

Finally there are modes (M) associated with the state of the system $m : S \rightarrow M$.

THE PROPERTIES TO BE VERIFIED OF THE MODEL

The properties that are used in the analysis have a common form. They are generally concerned with the effect or otherwise of a change in state attributes. Properties either specify something about *any* state transition that might take place, for example *visibility* specifies that a state change is visible: $filter(s_1) \neq filter(s_2) \Rightarrow p(filter(s_1)) \neq p(filter(s_2))$ where s_1 and s_2 are two consecutive states in the model, $filter$ extracts particular attributes and p selects a set of perceivable attributes relevant to the change. Often a property of this kind is constrained by a predicate $guard$. $guard$ restricts the states to those that are considered relevant to the particular property under consideration. For example, only states that are in a particular mode will lead to the property being true. Alternatively a property focuses on the behaviour of a particular *action* or set of actions. For example given that some pre-condition $guard(s)$ is true and the action a is permitted, $(per(a)(s) \wedge guard(s)) \Rightarrow (filter(a(s)) \neq filter(s))$.

The job of the analyst is to express these properties, choosing relevant filters and guards. These choices will lead to an understanding of where the interactive system might cause user confusion.

THE EXAMPLE

The chosen device (the Alaris GP infusion pump [3] — see Figure 1) has characteristics that are common to many devices that control processes. The clinician user sets infusion pump parameters and monitors the infusion process using the device. 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 at a pre-determined 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 alarms. When the 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 1). A subset of the features can also be changed when infusing. Chevron buttons are used to increase or decrease entered numbers incrementally. Depending on current mode they can be used to change infusion rate, volume to be infused and time, or alternatively allow the user to point at options for selection 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.

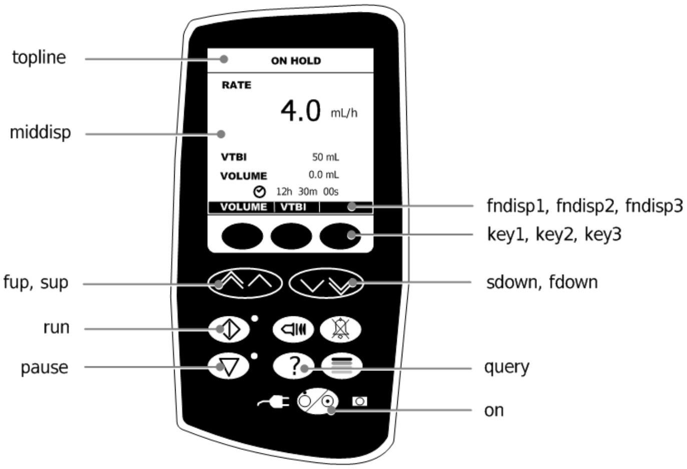

Figure 1. The pump user interface and actions

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.

THE PROPERTY TEMPLATES

The analysis approach uses templates to generate properties that are tailored to the device. As will be seen through example, expanding and tailoring the templates leads to insight about the device design as well as producing properties that will, if true of the design, lead to an interface that is more predictable and easy to use. The templates, to be described in more detail (derived from [2]) are: *completeness*, *feedback*, *consistency*, *visibility*, *reversibility* and *universality*. Informal descriptions of the heuristics, along with descriptions in the model, are described below. The use of one of the templates will be illustrated using the infusion pump example.

Completeness

This template is used to express that it is possible to reach any other state (or subset of states, for example mode) in one (or a few) steps. For example, being able to reach "home" from anywhere in one step is a completeness property.

The property template is expressed as always being possible to take action (a user action) from any state that satisfies a predicate $guard : S \rightarrow T$ to another class of states determined by a predicate $goal : S \rightarrow T$. The guard is introduced to make it possible to exclude states that may not be relevant. The property *completeness* is defined as: $\forall\, s \in S : guard(s) \wedge\!\sim goal(s) \Rightarrow \exists\, a \in A : goal(a(s))$.

Two modes are relevant in the case of the Alaris: *paused* and *infusing* where the chevron keys can be used to adjust the infusion rate, unless the infusion rate has previously been locked by the user. These are the normal default operating modes for the device.

Feedback

Feedback properties require that a change in the state (usually specific attributes of the state rather than the whole state) is visible or alternatively that an action always has an effect that is visible. The first version of the property is defined in terms of state changes where states satisfy a particular constraint $guard$ and the attributes that provide the focus for a particular instance of the property are specified by $filter$. The predicates and functions, as indicated in the example developed later, are instantiated with the details of the actual specification.

The formulations of the two properties are defined below. Note that there are two formulations of state feedback that depend on how the specification is structured (for example an attribute may simply be marked as visible in the specification or there may separately specified "perceivable" attributes).

state feedback (version 1) $(guard(s1) \wedge guard(s2) \wedge (filter(s1) \neq filter(s2))) \Rightarrow (p(filter(s1)) \neq p(filter(s2)))$

state feedback (version 2) $(guard(s1) \wedge guard(s2) \wedge filter(s1) \neq filter(s2)) \Rightarrow (VIS(filter(s1)) \wedge VIS(filter(s1)))$

action version $a : S \rightarrow S$ is such that for some $s \in S$ such that $per(a)(s)$ and $guard(s)$ then if, for some filtering function, $filter(s) \neq filter(a(s))$ then $VIS(filter(s)) \wedge VIS(filter(a(s)))$ and $p(filter(s)) \neq p(filter(a(s)))$

These properties are usually strengthened or weakened to enable proof as appropriate to context and safety requirements. Feedback properties for the Alaris device include:

- any action other than a chevron key changes the mode, and the change of mode is visible
- any chevron key always changes either a pump variable or cursor position in the menu, depending on the mode
- if a pump variable is changed then that change is visible (actually this property is more limited because the relevant variable in some cases is only visible after the change)
- if the mode changes then the mode is visible.

Consistency

Completeness and feedback properties describe consistent characteristics of the device. Further examples of consistency can be described of families of actions: $a \in A_c$ where $A_c \subseteq \wp(S \rightarrow S)$ and $per(a)(s)$ for some $s \in S$ then $consistent(a(s)) = consistent(s)$ where $consistent$ is a projection on S. The general characteristic of these properties is that an action (or group of actions) has similar effect. Examples include properties of actions that they *always* change mode, $a \in S \rightarrow S$ has the property that if for some $guard : S \rightarrow T$ such that $guard(s)$ and $per(a)(s)$ then $m(a(s)) \neq m(s)$. Alternatively a group of actions may have the common property that they *never* change mode: $a \in S \rightarrow S$ is *mode invariant consistent* if for some $guard : S \rightarrow T$ such that $guard(s)$ and $per(a)(s)$ then $m(a(s)) = m(s)$

These more specific mode related properties are also relevant to those basic attributes that specify the underlying process of the system that lies behind the interface (in the example, the infusion pump): if ba extracts the "basic attributes" of the device $ba : S \rightarrow B$ if $s \in S$ such that $guard(s)$, then for some action a such that $per(a)(s) \Rightarrow ba(s) = ba(a(s))$

Examples of these kinds of consistency in the Alaris are:

1. actions designated as function keys always change the entry mode
2. a chevron key will always change the pump variable indicated by the entry mode if the entry mode designates a pump variable (note that in some modes chevron keys are used to navigate the cursor)
3. when a function key is associated with a soft display of ok then the value of the relevant pump variable is changed to the value set within the entry mode
4. when a function key shows a soft display of $quit$ then the value set in the mode is discarded and the pump variable reverts to the value it had when it entered the mode.

Goal reversibility

An action can always be reversed. An action a is *reversible* if given a guard $guard : S \rightarrow T$, and for any $filter : S \rightarrow FS$ which extracts a set of focus attributes of the state, for any $s \in S : guard(s)$ there is a reverse action $b : S \rightarrow S$ such that $filter(a(b(s)) = filter(s)$

Visibility

For each $s \in B$ there corresponds a $pa \in P$ such that $VIS(s, pa)$. These properties specify an invariant relation between state attributes and display attributes. Examples of these properties are: the current entry mode is always unambiguously displayed; function key displays always appear the same in each entry mode; when entry of a particular device variable is ready then the value of that variable is visible.

Universality

Properties that specify an invariant relation between attributes of the state: for for example in a particular mode, the function keys will always show the same displays.

DEVELOPING A PROPERTY FOR THE ALARIS

A consistency property for the Alaris $key1$ action, when the function display for that key is "ok" is that it always reaches a state in which the top line of the display shows "holding", unless the infusion rate is 0. When the rate is 0 the top line shows "set rate". In terms of the model the goal of the proposed property is that

```
goal_ok(st) =
    (device(st)`infusionrate=0 &
     topline(st) = setrate) OR
       (device(st)`infusionrate \=0 &
        topline(st) = holding)
```

The guard is

```
preguard_ok(st) =
     (fndisp1(st) = fok) &
      NOT device(st)`infusing
```

This specifies that associated with key_1 there is a function key display which shows “ok”. It also specifies that the device must be paused for the property to be true. The property to be proved is that:

```
  okgoal_consistent(st: alaris) =
      (pre_ok(st) &
        preguard_ok(st)) =>
                goal_ok(st)
```

This property is proved using structural induction over all the possible actions permitted by the device (as specified by `alaris_transitions(pre,post)`):

```
okgoal_consistent_thm: THEOREM
    FORALL (pre, post: alaris):
        (init?(pre) =>
           okgoal_consistent(pre)) &
        ((alaris_transitions(pre,post) &
            okgoal_consistent(pre))
              => okgoal_consistent(post))
```

Attempting to prove the theorem fails, generating counter-examples indicating that the guard is not strong enough. The guard needs strengthening to exclude several modes. Through a process of repeated proof and counter-example discovery, a guard is developed that excludes this and other possibilities:

```
preguardok(st: alaris): boolean =
   fndisp1(st) = fok AND
   NOT device(st)`infusing? AND
   NOT ((topline(st) = options ) OR
        (topline(st) = vtbitime AND
        (entrymode(st) = vttmode)) OR
        (topline(st) = dispvtbi AND
          ((entrymode(st) = bagmode) OR
           (entrymode(st) = tbagmode))))
```

DISCUSSION AND CONCLUSIONS

Formal specification is challenging for non-experts. Producing the specification of the interactive system is complex. Once constructed it is then necessary to verify that the formal specification represents the implemented device accurately. Finally it is necessary to demonstrate that the device, as specified, satisfies the properties. This paper indicates how the specification of the interactive system may be simplified by structuring it: as user actions that transform states, distinguishing modes and perceivable attributes of the state. The problem of generating appropriate proofs becomes one of instantiating template properties to the specification and then using a theorem prover (PVS) to prove the properties are true of the specification. This is a relatively routine process. The formulation of theorems is much simplified, the interpretation of counter-examples generated by the automated prover is relatively straightforward with experience. The possibility of animating the formal models to create prototypes of the modelled interfaces, and the possibilities these prototypes raise in terms of discussing the results of verification with stakeholders, provides further opportunities for making these proofs clear. This approach is described in [5] and goes some way to help dealing with the problem of verifying that the formal specification is a realistic representation of the device.

ACKNOWLEDGEMENTS

José Creissac Campos and Michael Harrison were funded by project ref. NORTE-07-0124-FEDER-000062, co-financed by the North Portugal Regional Operational Programme (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese foundation for science and technology (FCT). Paul Curzon, Michael Harrison and Paolo Masci were 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.

REFERENCES

1. Arney, D., Jetley, R., Jones, P., Lee, I., Sokolsky, O., Ray, A., and Zhang, Y. Generic infusion pump hazard analysis and safety requirements. Tech. Rep. MS-CIS-08-31, University of Pennsylvania, February 2009.

2. Campos, J. C., and Harrison, M. D. Systematic analysis of control panel interfaces using formal tools. In *Interactive systems: Design, Specification and Verification, DSVIS ’08*, N. Graham and P. Palanque, Eds., no. 5136 in Springer Lecture Notes in Computer Science, Springer-Verlag (2008), 72–85.

3. Cardinal Health Inc. Alaris GP volumetric pump: directions for use. Tech. rep., Cardinal Health, 1180 Rolle, Switzerland, 2006.

4. Duke, D. J., and Harrison, M. D. Abstract interaction objects. *Computer Graphics Forum 12*, 3 (1993), 25–36.

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

6. Owre, S., and Shankar, N. A brief overview of PVS. In *Theorem Proving in Higher Order Logics, TPHOLs 2008*, O. A. Mohamed, C. Muñoz, and S. Tahar, Eds., vol. 5170 of *Lecture Notes in Computer Science*, Springer-Verlag (Montreal, Canada, Aug. 2008), 22–27.

7. US Food and Drug Administration. Infusion pump improvement initiative. Tech. rep., Center for Devices and Radiological Health, April 2010.

8. Woodcock, J., Larsen, P. G., Bicarregui, J., and Fitzgerald, J. Formal methods: Practice and experience. *ACM Computing Surveys 41*, 4 (2009), 19:1–19:36.