Analysing User Confusion in Context Aware Mobile Applications

K. Loer[1] and M.D. Harrison[2]

[1] Department Strategic Development, Germanischer Lloyd AG - Head Office, Vosetzen 35, 20459 Hamburg, Germany
loe@gl-group.com
[2] Correspondence address: Informatics Research Institute, University of Newcastle upon Tyne, NE1 7RU, UK
Michael.Harrison@ncl.ac.uk

Abstract. Mobility of ubiquitous systems offers the possibility of using the current context to infer information that might otherwise require user input. This can either make user interfaces more intuitive or cause subtle and confusing mode changes. We discuss the analysis of such systems that will allow the designer to predict potential pitfalls before the design is fielded. Whereas the current predominant approach to understanding mobile systems is to build and explore experimental prototypes, our exploration highlights the possibility that early models of an interactive system might be used to predict problems with embedding in context *before* costly mistakes have been made. Analysis based on model checking is used to contrast configuration and context issues in two interfaces to a process control system.

1 Background

Mobile interactive technologies bring new opportunities for flexible work and leisure. The fact that the mobile device is context aware means that user interaction can be more natural. The system (that is the whole software, human and hardware infrastructure) can detect where the device and its user are, infer information about what the user is doing, recognise urgency, even be aware of the user's emotional state. As a result user actions may be interpreted appropriately. The benefits that context awareness brings can be obscured by difficulties. Interaction may be confusing, surprising the user, and causing failure to occur.

Context aware systems are still mainly at an experimental stage of development and there is considerable interest in how these systems are used. The cost of user confusion about how action is interpreted may be expensive in terms of poor take up and potential error in safety critical situations. Techniques are required that can help predict these difficulties at design time. An important question therefore is what these techniques should be and whether the cost of using them is justified by the early understanding of design. The work underlying this paper uses formal modelling techniques and model checking. The point of using these techniques is not to suggest necessarily that an industrially scaleable

M.F. Costabile and F. Paternò (Eds.): INTERACT 2005, LNCS 3585, pp. 184–197, 2005.

technique should use them precisely as given in the paper nor that these techniques be used alone. It is instead the purpose to illustrate the type of issues that approaches such as these can help understand. An important question to be asked of any technique is whether early analysis requires a level of effort that can be justified in terms of potential costs of user confusions in business and safety critical systems.

The purpose here is to explore how analytic techniques might be used to:

- analyse differences between different interface configurations, in this case the difference between a central control room and a mobile hand-held PDA.
- analyse contextual effects. A simple model of context based on location is developed to analyse user action and user process.

The structure of the paper is as follows. The next section gives a scenario to illustrate the kind of system that is being considered here. Section 3 discusses the analysis to be performed. Section 4 presents briefly the model of the two user interfaces to the system. Section 5 explores the analysis of the system based on the models. The paper concludes by discussing the relevance of this approach and how future techniques might emerge.

2 A Scenario

A control room contains full wall displays on three sides. Plant schematics are displayed to represent the plant's state and can be manipulated through the control room interface using physical devices (e.g., switches), command line or direct manipulation interaction techniques, through the PDA interface, or through the physical components of the plant itself (e.g., closing a valve). Trend data about the plant is also displayed and helps operators anticipate emerging situations. Workflow information indicating today's schedule for an individual operator is contained in the operator's window also displayed on part of the wall.

A problem occurs in the plant requiring "hands-on" inspection and possible action from one or several operators. Operators (perhaps working as a team) take PDAs as they go to find out what has happened. General situation information and prompts about what to do next can be accessed from the PDA. The PDA can also be used to monitor and control a valve, pump or heater in situ (some of the monitoring characteristics of this device are similar to those described in [16]). A limited subset of information and controls for these components will be "stored" in the PDA to ease access to them in the future – analogous to putting them on the desktop. These desktop spaces are called *buckets* in [16]. The operator can view and control the current state of the components when in their immediate vicinity. Context is used in identifying position of an operator, checking validity of a given action, inferring an operator's intention, checking action against an operator's schedule assessing and indicating urgency.

For example, a leak in a pipe is indicated in the control room by a red flashing symbol over the relevant part of the schematic. Two operators walk out of the control room leaving it empty, one walks to the location of a heater downstream

of the leak, the other walks to the valve upstream of the leak. The operator upstream attempts to close off the valve using the PDA but is warned not to, while the other operator is told by the PDA that the heater should be turned off quickly because the first operator is waiting. Both operators, after having carried out their actions, put heater and pump status and controls (respectively) in buckets in their PDAs and move to the location of the leak to deal with it. When they have fixed the leak together they each check and restore the controls that they had previously put in buckets to the state before the leak was identified and walk back to the control room.

This scenario indicates the variety of modes and contexts that can occur. Confusions can arise if there is more than one plant component in close proximity, if the operator forgets which component they have saved, if one operator forgets that another operator is nearby. These problems can be exaggerated by poor design.

3 Analysing the Interface

Given a design such as the one above, it is clear that configuration and context are important to the success of the system. What happens to the interface when the operator moves from the control room to the handheld device and begins to move around the plant? What changes occur between the control room and the hand held device? How is the hand held device affected by the context? An operator will have a number of goals to achieve using these interfaces and the actions that are involved to do this will be different in the two interfaces, and in the mobile case dependent on context.

A typical approach to analysing these differences might be to perform a task analysis in different situations and produce task descriptions that can be used to explore the two interfaces and how the interfaces support the interactions. This might involve considering the information resources that would be required in the two cases [19]. Such an approach would have much in common with [17,7]. Indeed this analysis is performed in Loer's thesis [13]. However there are difficulties with such an approach. Task descriptions assume that the means by which an operator will achieve a goal can be anticipated with reasonable accuracy. In practice a result is that strategies or activities that the operator actually engages in may be overlooked.

A different approach is to take the models and check whether a goal can be reached *at all*. The role of a model checker is to find *any* path that can achieve a user goal. This new approach also has difficulties because the sequence of actions may not make any sense in terms of the likely actions of an operator. In order to alleviate this the analyst's role is to inspect possible traces and decide whether assumptions should be included about use that would enable sequences to be generated that are more realistic. The advantage of this approach is that it means that analysis is not restricted to sequences that are imposed – the presumed tasks. The disadvantage is that in some circumstances there may be many paths that might require such exploration.

A model of context is required, as well as of the devices, that will enable an analysis of the effects of the user interface of the mobile device in this way. Since the problem here is that action or sequences of actions (process) may have different meanings depending on context a clear definition of context is required. Persistently forgetting to restore information when the context has changed could be one effect of context, and can be considered as part of the analysis. In the case study the environment is described simply in terms of physical positions in the environment and transitions between these positions. As the hand-held device makes transitions it is capable of interacting with or saving different plant components onto the device. A model of the plant is included in order to comprehend how the interfaces are used to monitor and control.

Context confusions can be avoided through design by changing the action structure (for example, using interlocks) so that these ambiguities are avoided or by clearly marking the differences to users. Techniques are required that will enable the designer to recognise and consider situations where there are likely to be problems. The process is exploratory, different properties are attempted and modified as different traces are encountered as counter-examples or instances. Traces that are "interesting" are scrutinised in more detail to investigate the effectiveness of the design and the possibility of confusion – discovering an interesting trace does not of itself mean that the design is flawed or is prone to human error. Implications of different configurations are explored by considering simple assumptions about the user. In what follows we describe an experiment in which questions are articulated in LTL (Linear Temporal Logic) and recognised by the SMV model checker [15].

4 Modelling the User Interface

The characterisation of the device and of the control room are both much simplified for the purposes of exposition. The icons on the hand-held device are the only means available to the user to infer the current system state and the available operations. Since the visibility of icons is important to the operation of the plant and the usability of the hand-held device, the basis for the analysis is (i) that all available operations are visible, and (ii) that all visible operations are executable. The analysis uses Statecharts [9]: an example of how an interface can be developed using Statecharts is given in [11]. The Statecharts in the current scenario are structured into different components as suggested by [4] to make interaction with the device and the effect of the environment clearer and is based on a more detailed analysis described in [13].

The interactive system that controls the process is designed: (1) to inform the operator about progress; (2) to allow the operator to intervene appropriately to control the process; (3) to alert the operator to alarming conditions in the plant and (4) to enable recovery from these conditions. A model is required to explore usability issues and design alternatives in the light of these goals of the underlying process. The central control mechanism provides all information in one display (Section 4.1), while the personal appliance displays partial information (Section 4.2).

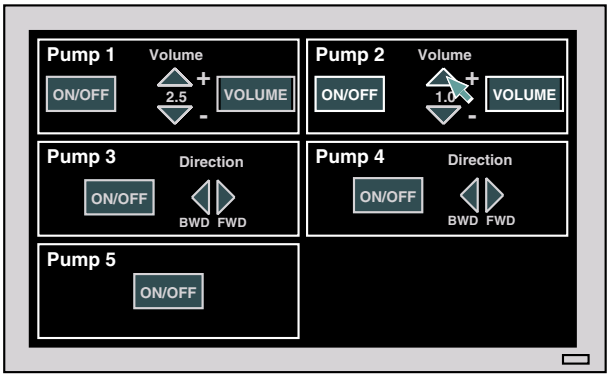

Fig. 1. Control Screen layout

4.1 Representing and Modelling the Central Panel

This paper deliberately glosses over the model of the process. The process involves tanks and pumps that feed material between tanks. The tanks can be used for more than one process and, in order to change processes, a tank must be evacuated before material can be pumped into it. In order to achieve this some of the pumps are bi-directional. In fact the process is expressed as a simple discrete model in which the significant features of the environment can be explored, for more details, see [14] or [2]. Hence the state of the tank is simply described as one element of the set $\{full, empty, holding\}$ — there is no notion of quantity or volume in the model. This is adequate to capture the key features of the process from the point of view of interaction with the system.

The control panel contained in the control room can be seen in Figure 1. All the pumps in the plant are visible and can be adjusted directly using a mouse. As can be seen from the display all the pumps can be switched on and off, some pumps (3 and 4) can be reversed and the volume of flow can also be modified in the case of pumps 1 and 2.

The control room, with its central panel, aims to provide the plant operator with a comprehensive overview of the status of all devices in the plant. Availability and visibility of action will be the primary concern here. Other aspects of the problem can be dealt with by using complementary models of the interface, for example alarms structure and presentation, but analysis is restricted for present purposes. The specification describes the behaviour of the displays and the associated buttons for pump 1 (and equivalently pump 2). The effects of actions are described in terms of the signals that are used to synchronise with the pump description and the states in which the buttons are illuminated.

The control panel is implemented by a mouse-controlled screen (see Figure 1). Screen icons act as both displays and controls at the same time. Hence from Figure 2 we can see that `PUMP1USERINTERFACE` supports four simple on-off state transitions defining the effect of pressing the relevant parts of the display. The state indicates when icons are illuminated but also shows that the actions trigger

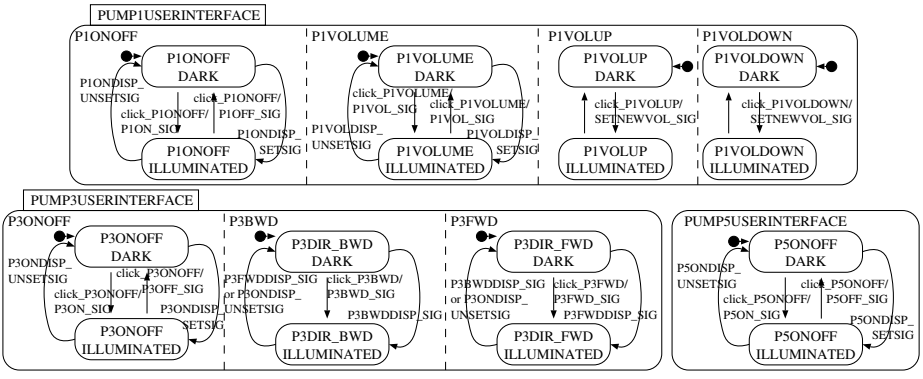

Fig. 2. Initial specification of control screen behaviour

corresponding actions in the underlying process. The Statechart here builds a bridge between actions that relate to the behaviour of the process underneath and actions that correspond to using the mouse to point and click at the relevant icons. A detailed account of what the specification means is not presented here. An indication of what it would look like is all that is intended at this stage – an indication of the scale of the modelling problem using this style of specification. Many other approaches could have been used: Paternò used LOTOS [17], Campos and Harrison used MAL [2]. Notations such as Promela that are supported directly by model checkers are also relatively straightforward to use [10].

4.2 Representing Context and the Hand-Held Control Device

The hand-held device uses individual controls that are identical to those of the central control panel but only a limited amount of space is available for them. As a controller walks past a pump it is possible to "save" the controls onto the display. Thereafter, while the controls continue to be visible on the display, it is possible to control the pumps from anywhere in the system.

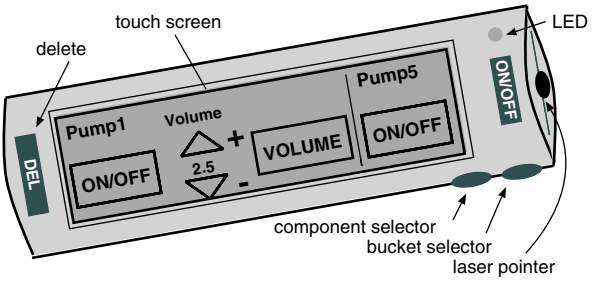

Fig. 3. A hand-held control device (modified version of the "Pucketizer" device in [16])

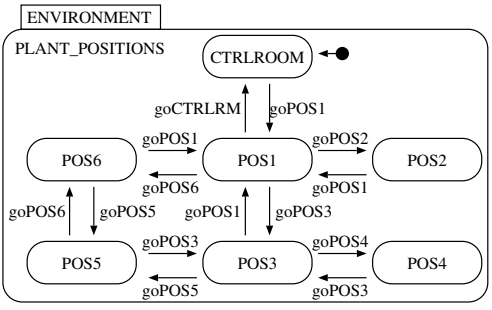

Fig. 4. Model of device positions

The hand-held control device (Figure 3) knows its position within the spatial organisation of the plant. Hence the ENVIRONMENT model to describe the system involving this device is extended to take account of context. A simple discrete model describes how an operator can move between device positions in the plant modelled as transitions between position states, as shown in Figure 4.

By pointing the laser pointer at a plant component and pressing the component selector button, the status information for that component and soft controls are transferred into the currently selected bucket. Components can be removed from a bucket by pressing the delete button. With the bucket selector button the user can cycle through buckets. The intended use of the device has been altered from the description contained in [16] from monitoring and annotating to monitoring and manipulation.

The specification of the hand-held device describes both the physical buttons that are accessible continuously and other control elements, like pump control icons, that are available temporarily and depend on the position of the device. When the operator approaches a pump, its controls are automatically displayed on the screen (it does not require the laser pointer). The component may be "transferred" into a bucket for future remote access by using the component selector button. Controls for plant devices in locations other than the current one can be accessed remotely if they have been previously stored in a bucket. When a plant component is available in a bucket and the bucket is selected, the hand-held device can transmit commands to the processing plant, using the pump control icons.

Figure 5 shows an extract of the specification. Here the user can choose between three buckets and each bucket can store controls for up to two components. In the `BUCKETS` state the current contents of each bucket x are encoded by variables "Bx`CONTENT`".

The environment in this case is a composition of the tank content model and the device position model in Figure 4. The model presumes that the appliance should always know its location. This is of course a simplification. Alternative models would allow the designer to explore interaction issues when there is a dissonance between the states of the device and its location. A richer model in

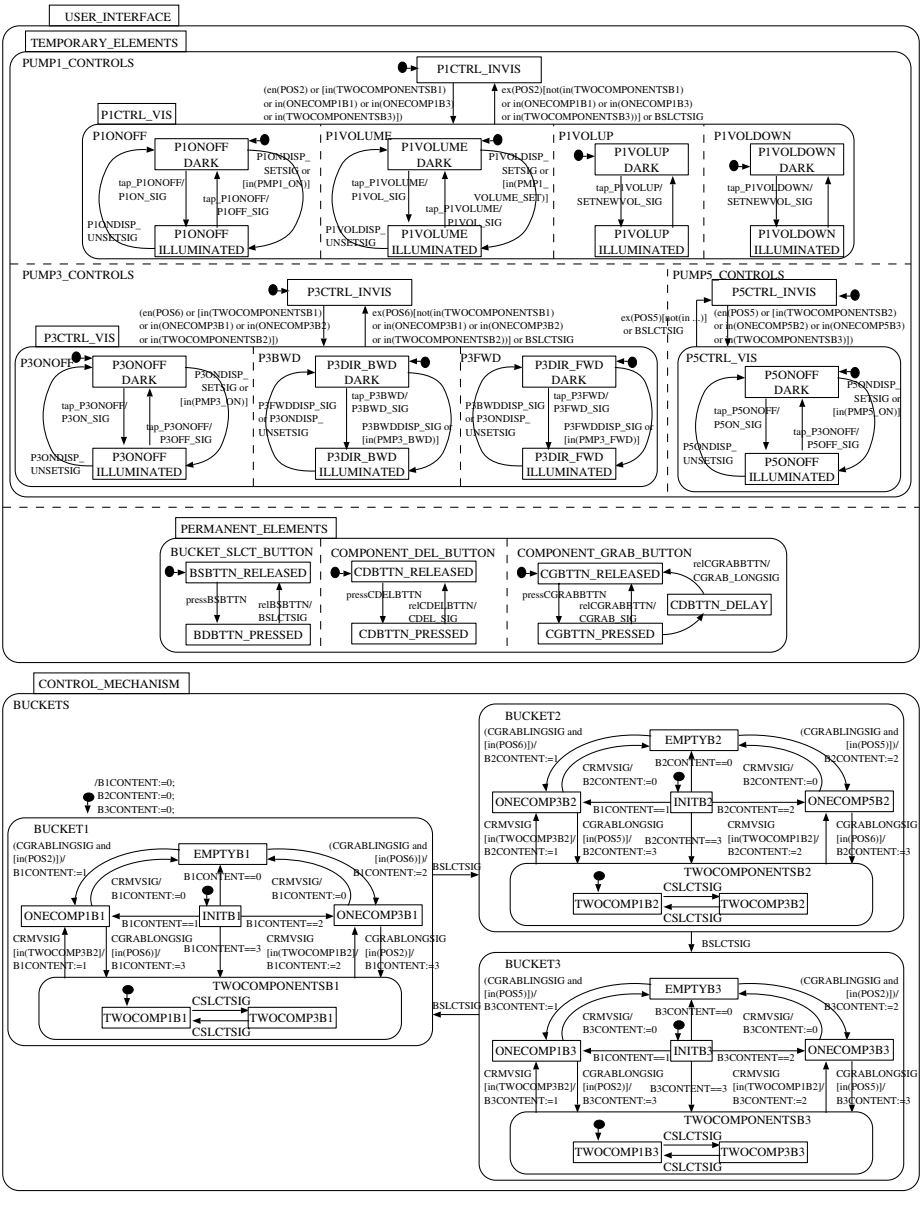

Fig. 5. OFAN model for the hand-held device: The USER INTERFACE and CONTROL MECHANISM modules

which variables are associated with states, and actions may depend on values of the state that have actually been updated, may lead to asking questions of the models as whether "the action has a false belief about the state". These issues are important but are not considered in this paper.

5 Analysis

Model-checking is a technique for analysing whether a system model satisfies a requirement. These requirements may be concerned with a number of issues including safety and usability. The model checker traverses every reachable system state to check the validity of the given property. If the property "holds", a `True` answer is obtained. Otherwise, the property is `False`, and the tool attempts to create a sequence of states that lead from an initial state to the violating state. These "traces" are a valuable output because they help understanding *why* a specification is violated. There are many detailed expositions of approaches to model checking, see for example [3,12,1,10] and a number of treatments of interactive systems from a model checking perspective, see for example [17,7,2,18].

5.1 Comparing the Control Room and the Hand Held Device

In order to explore the effect of the difference between the control room and the hand-held device a reachability property may be formulated for a user level "goal" of the system. The goal chosen here for illustration is "*Produce substance C*" which is a primary purpose of the system.

The idea is that differences are explored between the traces by two models: on the one hand containing the control room interface; on the other hand containing the mobile device. If a property does not hold then the checker finds one counter-example. Alternatively, the negated property may be used to find a trace that satisfies the property. Usually the model checker only produces a single trace giving no guarantee that it is an interesting one from the point of view of understanding design implications. Additional traces can be created by adding assumptions about the behaviour. This contrasts with an approach using explicit tasks (see for example, [7,13]) where the model checker is used to explore a particular way in which the goal can be achieved (the task). So far as this paper is concerned any behaviours required to achieve a goal are of interest.

The sequences in Figure 6 are visualisations of the traces obtained by checking for different models if and how the plant can deliver substance C to the outside world. The property asserts that, eventually, pump 5 will be turned on with tank 1 holding substance C. This is specified as:

```
SAN1:
    F (PUMP5CTRLM.state=PMP5ON)
      & (TANK1.state = HOLDS_C)
```

In this case the negated property "`not SAN1`" is used because instances that satisfy the property are required. The two models involving the different interfaces are checked with the same property. The first sequence in Figure 6 satisfies the control room interface. The second sequence was generated by checking the property against the hand-held device model. While the first two traces assume a serial use of pumps, the third and fourth sequences show the same task for a concurrent use of pumps. Comparison of these sequences yields information about the additional steps that have to be performed to achieve the same goal.

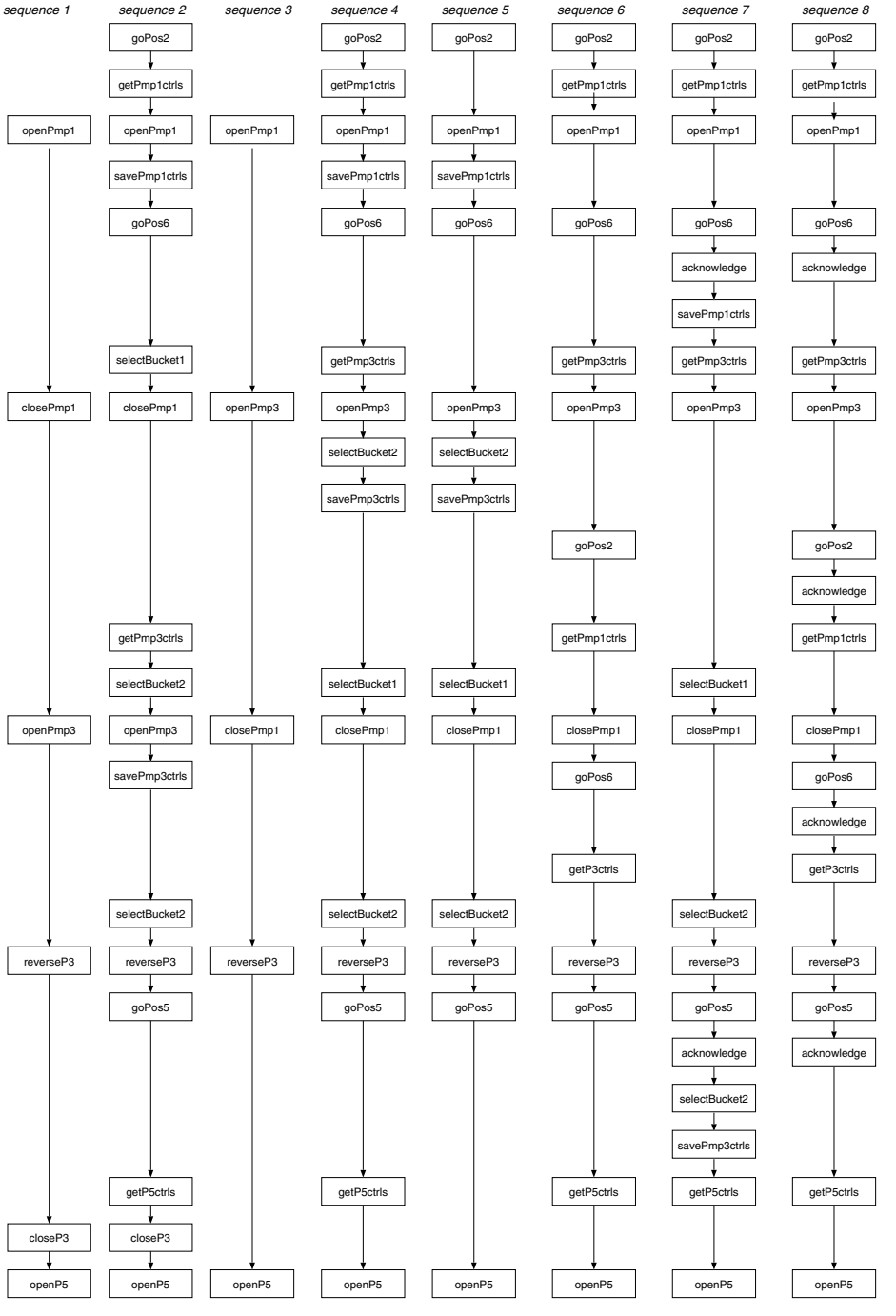

Fig. 6. Traces generated by runs of the model checker

5.2 Analysing Context Effects

As a result of making a comparison between the traces for the control room and for the hand held, the analyst might come to the conclusion that the repetitive process of saving controls may cause slips or mistakes, a direct effect of location on the actions of the hand-held device. To explore the effect of this a further assumption may be introduced to the property to be analysed, namely that an operator might forget certain steps.

This assertion "`alwaysForget`" which states that controls for any of the pumps are never saved is described as follows:

```
assert alwaysForget:
 G !(savePmp1ctrls| [...] |savePmp5ctrls);
```

The original property `SAN1` is checked under the assumption that this assertion holds:

```
assume alwaysForget;
 using  alwaysForget prove SAN1;
```

Checking this property leads to the sixth sequence in Figure 6. A consequence of exploring this sequence highlights the likelihood of context confusions and therefore the need for the redesign of the device. As can be seen, an identical subsequence of actions at positions 2 and 6 have different effects. An interlock mechanism is therefore introduced with the aim of reducing the likelihood that human error arising from forgetfulness might arise. The proposed redesign warns the user and asks for acknowledgement that the currently displayed control elements are about to disappear.

The warning is issued whenever a device position is left and the device's control elements are neither on screen nor stored in a bucket. It is straightforward to adjust the model of the interface to the hand-held device to capture this idea, and this specification is given in the fuller paper [14]. The design however does not prevent the user from acknowledging and then doing nothing about the problem.

Checking the same properties, including the assumptions about the forgetful user, produces Sequences 7 and 8 in Figure 6. In this example the central control panel characterises the key actions to achieving the goal since the additional actions introduced by the hand held device are concerned exclusively with the limitations that the new platform introduces, dealing with physical location, uploading and storing controls of the visited devices as appropriate. The analysis highlights these additional steps to allow the analyst to subject the sequence to human factors analysis and to judge if such additional steps are likely to be problematic. The reasons why a given sequence of actions might be problematic may not be evident from the trace but it provides an important representation that allows a human factors or domain analyst to consider these issues. For example, action `goPOS6` may involve a lengthy walk through the plant, while action `savePmp4ctrls` may be performed instantaneously and the performance

of action `getPmp3ctrls` might depend on additional contextual factors like the network quality. The current approach leaves the judgement of the severity of such differences to the designer, the human factors expert or the domain expert. It makes it possible for these experts to draw important considerations to the designer's attention.

6 Conclusions

The paper illustrates how configuration and context confusions might be analysed in the early stages of design before a system is fielded. We emphasise again that the exploration of these techniques makes no presumption that these would be the only techniques used to explore potential user confusions. The particular method described involves comparing and inspecting sets of sequences of actions that reach a specified goal state. No assumptions are made about user behaviour initially, constraints based on domain and user concerns are used to explore subsets of the traces that can achieve the goals. Experts assist the process of adding the appropriate constraints to the properties to be checked. In order to do this a human factors expert or a domain expert may be provided with sufficiently rich information that it is possible to explore narratives surrounding the traces generated.

Hence traces can form the basis for scenarios that aid exploration of potential problems in the design of mobile devices, e.g. the additional work that would be involved for the system operator if subtasks are inadvertently omitted in achieving the goal. The tool can also be used to find recovery strategies if an operator forgets to store control elements.

Further work is of course needed to devise strategies for appropriate guidance with respect to (i) finding an efficient sequence of analysis steps and (ii) devising a strategy for the introduction of appropriate assumptions. Guidance is also required to help limit the size of the models to be analysed. Suitable techniques and heuristics for semantic abstraction of system models need to be devised to avoid the state explosion problem. However, the size of models that can be dealt with is encouraging and this situation can be improved through appropriate abstraction and consistency checking.

As has been said the case described in the paper involves an oversimplistic model of context for the purpose of presentation. The following questions require exploration:

- What are the key features of the design that are relevant to these context confusions? In the work described here the further step of evaluating whether the properties that are analysed actually cause user confusion is assumed to be carried out by a human factors expert who would assess the traces generated by the technique.
- What are appropriate models of context — what about the information that might be inferred at these different positions? What about knowledge about history or urgency? What about the proximity, knowledge and behaviour of other mobile agents in the environment? What about issues such as the

staleness of data? A number of papers [5,8] classify and critique notions of context.
- If more than one model is appropriate, at different stages of the design or at the same time, how are these different stages and complementary models used together?

More elaborate analysis would involve models of context in which other users and configurations (for example PDAs) may enter or leave dynamically. In order to reason about context such as these, knowledge logics using operators such as the K operator could be used to express what an agent knows in a given context [6]. Since K-logic is described in terms of a Kripke model it is relatively straightforward to perform model checking using it. Hence given the scenario example, a question may be asked such as whether it is common knowledge that the repair has been completed in order that all agents can restore the state of the components they were dealing with to their original states. The model and logic may also be used to ask whether it is possible that an agent can think that the state of their component can be restored before it is time to do it. Hence the logic will be used to express properties that capture potential user confusions in this richer notion of context.

With appropriate models and notions of user context confusion, it becomes possible to consider the pragmatics of modelling and analysis using these techniques. Similar strategies may also be adopted for exploring other aspects of context confusion, for example exploring the significance of the temporal validity of the state of a *bucket* on the user's ability to achieve goals within different timescales.

Acknowledgements

This work was supported by BAE Systems, the EPSRC DIRC project GR/N13999 and dstl.

References

1. M. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen. *Systems and Software Verification. Model-Checking Techniques and Tools.* Springer, 2001.
2. J.C. Campos and M.D. Harrison. Model checking interactor specifications. *Automated Software Engineering*, 8:275–310, 2001.
3. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking.* MIT Press, 1999.
4. A. Degani. *Modeling Human-Machine Systems: On Modes, Error, and Patterns of Interaction.* PhD thesis, Georgia Institute of Technology, December 1996.
5. A.K. Dey, G.D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16:97–166, 2001.
6. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, 2004.

7. 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.
8. J. Grudin. Desituating action: digital representation of context. *Human-Computer Interaction*, 16:257–268, 2001.
9. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.
10. G.J. Holzmann. *The SPIN Model Checker, Primer and Reference Manual.* Addison Wesley, 2003.
11. Ian Horrocks. *Constructing the User Interfaces with StateCharts.* Addison Wesley, 1999.
12. M. R. A. Huth and M. D. Ryan. *Modelling and reasoning about systems.* Cambridge University Press, 2000.
13. K. Loer. *Model-based Automated Analysis for Dependable Interactive Systems.* PhD thesis, Department of Computer Science, University of York, UK, 2003.
14. K. Loer and M.D. Harrison. Analysing and modelling context in mobile systems to support design. `http://homepages.cs.ncl.ac.uk/michael.harrison/publications.htm`, 2004.
15. K.L. McMillan. *Symbolic model checking.* Kluwer, 1993.
16. J. Nilsson, T. Sokoler, T. Binder, and N. Wetcke. Beyond the control room: mobile devices for spatially distributed interaction on industrial process plants. In P. Thomas and H.-W. Gellersen, editors, *Handheld and Ubiquitous Computing, HUC'2000*, number 1927 in Lecture Notes in Computer Science, pages 30–45. Springer, 2000.
17. F. Paternò and C. Santoro. Support for reasoning about interactive systems through human-computer interaction designers' representations. *The Computer Journal*, 6(4):340–357, 2003.
18. John Rushby. Using model checking to help discover mode confusions and other automation surprises. *Reliability Engineering and System Safety*, 75(2):167–177, February 2002.
19. P.C. Wright, R.E. Fields, and M.D. Harrison. Analyzing human-computer interaction as distributed cognition: the resources model. *Human-Computer Interaction*, 15(1):1–42, 2000.