Extending and evaluating a pattern language for safety-critical user interfaces

Simon Connelly* **Jay Burmeister†** **Anthony MacDonald*†**

Andrew Hussey‡

*Software Verification Research Centre,
The University of Queensland,
Brisbane, Qld 4072, Australia.
simonc@svrc.uq.edu.au

†School of Computer Science and Electrical Engineering,
The University of Queensland,
Brisbane, Qld 4072, Australia.
{jay, anti}@csee.uq.edu.au

‡Silber Software,
Sweden
andrew.hussey@silbersoftware.com

Abstract

This paper describes the extension and evaluation of Hussey's pattern language for safety-critical user interface development [Hussey and Mahemoff, 1999]. The patterns were updated and new patterns were added by examining existing medical systems. The new patterns were found by generating problem definitions from perceived exemplary solutions. This method of pattern development and its application is described. A qualitative evaluation of the pattern language was performed by designing a safety-critical user interface for a hypothetical radiation therapy machine. Discussion of the design process focuses on the usefulness of our patterns as a design tool for safety-critical user interfaces.

1 Introduction

Most safety-critical systems require that a human operator be involved in their operation. Controls are needed within a system to prevent or inhibit the hazardous effects introduced by human operators. Poorly designed user interfaces may lead to situations where a user's action inadvertently has hazardous consequences. Leveson [Leveson, 1995] discusses many occurrences of poor interface design leading to catastrophic incidents. We describe a pattern language that enables designers to reduce the occurence of accidents that are due to operator error. The pattern paradigm [Alexander *et al.*, 1977, Gamma *et al.*, 1994] provides a useful method for presenting solutions to problems identified during the design phase of the software life cycle. This paper focuses on the usefulness of design patterns when applied to safety-critical user interfaces, in particular medical systems. The main aim of this paper is to demonstrate how safe user interfaces can be designed by combining usability principles and safety.

Hussey's pattern language (referred to as the "initial" patterns or pattern language in this paper)[Hussey and Mahemoff, 1999], was examined, gauging its effectiveness and noting any inconsistencies. A collection of exemplary safety-critical user interfaces from the medical domain was gathered to extract "good" patterns from those solutions. By applying the initial patterns to these exemplary interfaces, we were able to assess their utility and add patterns that provided solutions to problems that were not addressed by the initial pattern language. This process for generating new patterns was similar to reversal of the intervention process from Human Reliability Assessment (HRA) [Kirwan, 1990] and is discussed in Section 3.2. We also replaced the examples used within the initial language with these "good" examples to produce a pattern language for medical system user interfaces.

Section 2 provides a brief overview of the relevant literature pertaining to patterns, usability, and safety-critical issues. The justifications for using a pattern-based approach to design rather than using guideline documents is also provided in Section 2. Section 3 presents a detailed description of our modifications and additions to the pattern language developed by Hussey. Section 4 describes the safety-critical issues related to the user interface for a hypothetical radiation therapy machine. The design process described provides a validation of our additions and modifications to Hussey's pattern language, and a method of ascertaining whether patterns prove useful for designing safer interfaces. Section 5 summarises our findings and conclusions.

2 Background

This section provides a brief introduction to safety-critical usability principles and patterns (design and usability). Safety-critical usability principles are outlined in Section 2.1, along with their relevance to safety-critical user interfaces. Section 2.2 introduces design patterns, and discusses why they are useful tools in the design of user interfaces for safety-critical system. Section 2.2 also introduces usability and safety-critical usability patterns.

2.1 Safety-critical usability

Usability principles can be used to guide the design of user interfaces for safety-critical computer systems. However, when a designer considers these principles in a safety-critical interface, there will be a different focus than in a "normal" system. There are six usability properties that an interface designer should take into account [Hussey and Mahemoff, 1999] when designing a user interface:

- Robustness: the likelihood of user error and the ability to recover from errors.
- Task efficiency: a measure of how efficient the interaction will be.
- Reusability of knowledge: an evaluation of how the new interface relates to the user's knowledge base.
- Effectiveness of user-computer communication: the level of understanding the user acquires from the use of the interface.
- Flexibility: the ability of the interface to adapt to new situations.
- Consistency: an evaluation of how consistent an interface's behaviour is.

In a safety-critical context, a designer will need to apply all of these properties; only the level of application will differ. For example, a safety-critical system need not be as efficient as a normal system, because making a system efficient can lessen the user's view of how important the interactions are. In all situations the main consideration of the designer will be the property of robustness, due to the hazardous and complex nature of safety-critical systems. If a user interface is robust there is less likelihood of an operator error causing a hazard state.

The flexibility and task efficiency properties will be the two most likely to be traded off against robustness. Flexibility is likely to be lessened because in a safety-critical user interface, a user should not be able to customise the interface to any large degree. This is due to the confusion customisation could cause in an emergency. Confusion is likely as a user is trained for "disaster recovery" in a certain interface configuration and if a hazard arises when the interface is configured differently to the expected manner, vital time can be lost, and misdiagnosis of the problem is more likely. In general, task efficiency is less important than robustness to safety-critical user interfaces where it is more important that a task be performed correctly (possibly through repeated entries of the same information) than efficiently. However, in time-critical operations such as those encountered in air-traffic control, efficiency is of equivalent importance to robustness.

Effective user-computer communication and reuse of knowledge are important within an interface as they aid in its robustness. For a user to maintain an accurate mental model of the system state, the interface should communicate all changes effectively and quickly.

2.2 Patterns

Design patterns provide an intermediate level of principles between universal guidelines and highly specific style guides. A style guide for user interfaces [Open Software Foundation, 1993] is a document that outlines how a system should look when it is completed; there is little guidance as to how the system should be designed to achieve this. An user interface standard [Apple Computer, 1987, IBM, 1992, Microsoft, 1995, Sun Microsystems, 1989] is a document that specifies the mechanics of how an interface behaves; for example, it may specify that at each step of a menu, it should be no more than 4 levels deep.

User interface style guides and standards have come under criticism for several reasons:

- difficulty of guideline interpretation,
- too simplistic and unable to be used effectively by people who are not human factors experts, and
- excessive effort required to find relevant sections.

The result is that user interface guides and standards often lead to poor and diverse interpretation, because designers are often inexperienced in anticipating Human Factors issues. Even if guidelines are more specific, they are still open to interpretation by designers. Additionally it is difficult for designers to decide which guidelines to use in a specific context.

In their paper, "Principles for a Usability-Oriented Pattern Language", Mahemoff and Johnston [Mahemoff and Johnston, 1998] state:

> "An approach to usability based on design patterns enables designers to learn how certain recurring problems can be solved according to high-level principles. "

Design patterns are a convenient way to help develop abstract design rules in the context of concrete projects.

Patterns were originally developed by Alexander and presented in his book "A Pattern Language" [Alexander *et al.*, 1977]. Alexander proposed using his patterns [Alexander, 1979] to solve town planning and architectural (building) design problems. His patterns covered recurring design problems and how they were solved by analysing the forces to be resolved, and exposing a design solution that satisfactorily resolved them. Once a pattern has been applied, more complex forces will come into effect, and thus more patterns will need to be applied. The set of patterns used to solve all problems is called a pattern language, which is formed when a collection of patterns is arranged in a cohesive structure. In a pattern language, higher-level patterns yield contexts that are resolved by more detailed patterns.

The aim of a pattern language is to capture the patterns of design solutions. By specifying the patterns that a solution follows, a designer will be able to extend a solution to apply to other systems. Pattern languages do not provide immediate solutions to problems. They do, however, accelerate a designer's knowledge acquisition, by demonstrating how design problems may be solved.

Gamma later applied Alexander's theories to software design in the book "Design Patterns" [Gamma *et al.*, 1994]. Mahemoff and Johnston [Mahemoff and Johnston, 1998] extended pattern-based design to user interface usability. Hussey applied the concept of interface patterns to safety-critical systems user interfaces.

Each pattern has certain attributes and Hussey's patterns have the following attributes: *Name*, *Context*, *Problem*, *Forces*, *Solution*, *Examples*, *Design issues*, and *Resulting context*. The attributes, *Forces* and *Examples*, are introduced below. For detailed explanation of these attributes, except *Forces* and *Examples*, see Hussey [Hussey and Mahemoff, 1999].

Forces are principles that influence the decision-making process when a designer is confronted with the problem in the context associated with the pattern. Forces may be contradictory. This means that there may be no single solution to a design problem. Pattern solutions aim to take into account as many of these forces as possible.

Examples are real-life examples of successful system features that embody the solution. Since considering relevant examples from situations not directly concerned with safety-usability can enhance the validity of the pattern, some pattern examples are from non-safety-critical interfaces. However, each pattern contains at least one example from a safety-critical system.

2.2.1 Safety-critical usability patterns

Hussey [Hussey and Mahemoff, 1999] focused Mahemoff's usability patterns onto safety-critical user interfaces. The pattern language covers safety-usability, in particular the effect of applying robustness enhancing principles (detailed in Section 2.1) on the usability of a safety-critical system. The main aim of producing this pattern language is focused on making an interface as usable as possible, whilst maintaining its inherent safety. Hussey and Atchison [Hussey and Atchison, 2000] discuss user-interface principles that are found in several major safety standards [Commonwealth of Australia, 1998, International Electrotechnical Commission, 1997].
The principles discussed are similar to those dealt with in the patterns described in this paper. Reference to the safety standards could be combined with the pattern solutions to provide a more complete resource for designers.

Hussey separated the pattern language into four categories: Task management, Task execution, Information and Machine control. The patterns were separated into the 4 categories to aid the designer in choosing which patterns to apply in a given situation and to increase the ease with which a pattern is found. Hussey provided fourteen patterns and these are listed below in their categories.

These patterns aim to provide a start point for a designer. It is envisaged that a designer will produce their own patterns in time, modifying the existing language (presented in Appendix A).

Task management The control flow of the human computer interaction.

Includes the following patterns: Recover, Stepladder, Task Conjunction, and Transaction.

Task execution The physical mechanisms through which users perform tasks.

Includes the following patterns: Affordance, Separation, Distinct Interaction, Preview, and Behaviour Constraint.

Information The information that is presented to users, and how it is presented.

Includes the following patterns: Reality Mapping, Abstract Mapping, Redundant Information, Trend, Interrogation, and Memory Aid.

Machine control The provision of system level functions that removes the responsibility for correct system behaviour from the user, and places it onto the system.

Includes the following patterns: Interlock, Automation, Shutdown, and Warning.

3 Extending the pattern language

Currently, extending pattern languages occurs on an ad-hoc basis. In this paper a more methodical approach is taken as introduced in Section 3.2. One aspect of this methodical approach is a reliance on exemplary user interfaces; criteria for these are given

in Section 3.1. The new and modified patterns are introduced in Section 3.3.

3.1 Example interface criteria

> "There is no guarantee that showing someone how something was performed incorrectly means that they will be able to infer how to do it correctly" [anon]

The criteria used for selecting example interfaces for this paper were that they:

- be safety critical systems,
- be well documented,
- have documentable design solutions, and
- have no documented failures within the chosen portion of the user interface.

Only those example interfaces that were seen as exemplary were chosen. Finding good example interfaces proved to be harder than first expected, as most safety-critical systems literature focuses on the negative points of an interface. All of the chosen examples[1] were from the medical field due to the availability of positive documentation on medical systems. An added benefit of all of the example interfaces being within the same field was increased consistency when providing examples for each pattern.

3.2 Pattern discovery method

Testing an existing pattern language against a group of exemplary user interfaces is a relatively straightforward task. However, discovering new patterns and solving problems within existing patterns is a time consuming and difficult task. The patterns in this paper underwent multiple iterations before they were complete. This section details how the initial patterns were modified and how the new ones were synthesised.

The first step in the process involved collating a library of exemplary user interfaces that satisfied the requirements described in Section 3.1. This library was collated through a detailed review of the safety-critical systems literature and through visits to medical institutions to view safety-critical user interfaces. The set of exemplary interfaces was then examined to see if they displayed the patterns contained in Hussey's initial language.

During this phase many minor problems and inconsistencies within the initial pattern language were found and corrected. These problems were generally improper *Force* resolution. The *Examples* within the initial language were also replaced with examples from the collected exemplary interfaces. This served to update the *Examples* and to replace those that did not adequately capture the solution.

[1] The chosen example interfaces are presented in Appendix B of [Connelly *et al.*, 2001].

The second step sought those facets of the interfaces that had not been captured in the initial pattern language. This involved looking at the documentation for a safety-critical system and looking for possible design interventions. These design interventions were then assessed to ascertain the solution that a particular intervention was trying to represent. Assessment of the problem a particular solution was focusing on was the final aspect of this second step.

The process of pattern discovery involved performing some steps similar to the intervention process from HRA (Human Reliability Assessment [Kirwan, 1990]) in reverse. Starting from a solution (the exemplary user interfaces) the nature of the problems that they were solving was ascertained. Matching a problem to a solution highlighted the context within which it had been used. The most difficult part of creating a pattern lies in defining the forces that affect the solution.

The final phase of documenting a pattern is defining the *Resulting Context* and noting any design issues that may be associated with applying the pattern, including problems that could arise from the application of the pattern and related patterns. If there were no existing patterns dealing with the resultant problems, a new pattern should be developed to solve the problem. An overview of the patterns that resulted from this analysis is presented in Section 3.3.

One major modification that was made to the structure of the initial pattern language relative to Hussey's pattern language was the modification of the categorical placement of the patterns within the language. In Hussey's language a pattern could only fit into a single category and this overly constrained the definition of the patterns. The new categorising system allows a pattern to belong in more than one category, if it contains elements of each. Related to this changed categoricsl placement was a new method of presenting the language. An index to the patterns at the start of the language was provided, with the patterns listed in their respective category/categories, and to provide an alphabetical listing of the patterns after the index[2].

3.3 New and modified patterns

The group of new patterns synthesised via the method outlined in Section 3.2 are detailed in Section 3.3.1. The patterns that are modifications of existing patterns are described in the Section (3.3.2). A description of these patterns is included in Appendix A.

3.3.1 New patterns

Accessibility When designing a system, the designer must take into account the cultural and physical abilities of the entire set of intended users. The Accessibility pattern is one of the more interesting

[2] As shown in Appendix A of [Connelly *et al.*, 2001].

patterns that was synthesised due to its placeholder nature. An entire sub-set of patterns would be required to solve the problems presented by the Accessibility pattern.

Automation Feedback When an automated system is running, it should provide feedback methods that will allow users to maintain their mental model of the system state.

Proximity When a system involves dangerous components, there is the possibility that it may harm a user. Where possible, a user should be physically separated from these dangerous components.

Training There are two training patterns: Training for uncontrolled environments, and Training for uncontrolled interfaces. Each provides similar solutions, for slightly different contexts. The training patterns suggest that if there are portions of a system that are uncontrollable for the designer they should provide methods for training users to reduce the risk of the system. These methods should provide the users with a high fidelity simulator of the system if the real system cannot be removed from operation during training [Bainbridge, 1987].

It is important to note that these patterns are designed for use when no other methods of risk mitigation are possible.

The difference between the two patterns is the contexts in which they are used. The uncontrolled environment pattern is applied to a system when the risks associated with the environments in which it will be operated cannot be controlled. On the other hand, the uncontrolled interface pattern deals with those systems that have had the underlying functional behaviour changed, but the interface remains essentially unchanged.

3.3.2 Modified patterns

Alert When the system enters into a hazard state and the system cannot handle the exception through Machine control, it becomes necessary to initiate an Alert. The user's attention should be directed to the disturbed parameters within the system via noise and graphical representations.

Notification All important changes of state within a system be brought to the user's attention via an audible event and/or a corresponding display of the state change.

4 Case study

The intention of this case study was to use the system specifications provided in Section 4.1 and the patterns developed from Hussey's initial pattern language to design a safe system. This system also needed to satisfy the usability principles outlined in Section 2. Safety considerations for the system are introduced in Section 4.2. Through the production of this case study (detailed in Section 4.3) we hoped to identify any problems or inconsistencies within the modified pattern language. The application of patterns in the design of the system is outlined in Section 4.4. Finally, a brief summary of the case study is given in Section 4.5.

4.1 Functional Requirements

The specification outlined below is taken from an experiment run by Mahemoff [Mahemoff, 2000]. When reading the requirements for this system it is important to realise that they were not written by a medical domain expert, and as such the system is provided as an example specification from which to build an interface.

The system is a radiation therapy application, designed to deliver measured amounts of radiation into a patient's brain via two track-mounted lasers around the patient's head. The patient's head is fixed in a stationary position; the two lasers are then calibrated to intersect at the area within the patient's head requiring treatment. The intersection between the lasers is the point of highest energy concentration, and hence the area in which the radiation is powerful enough to treat the patient. Although each laser alone is not strong enough to cause immediate effect, prolonged exposure to a single beam could be harmful to the patient.

The two lasers are mounted on a single track with a rotation range of 360 degrees. The lasers can also change the vertical angle at which they will point. Information about each laser is entered as a set of parameters:

- θ (0-360 Degrees): the angle around the track.
- γ (0-360 Degrees): the angle indicating the direction of the laser.
- w (0.0-100.0μm): the beam width.
- t (0.0-600.0s): the time period of the treatment.

The system also contains an optical motion sensor to detect if the patient's head moves during the treatment. In addition to this motion sensor there is also a heartbeat monitor attached to the patient, capable of sampling the patient's heart rate at five-second intervals.

4.2 Safety considerations for the system

There are many safety-critical issues that apply to this system that may need to be considered in designing the interface. These issues are listed below:

- Incorrect usage of the system could lead to severe injury to the patient, for example, excess radiation treatment may damage a patient's brain.

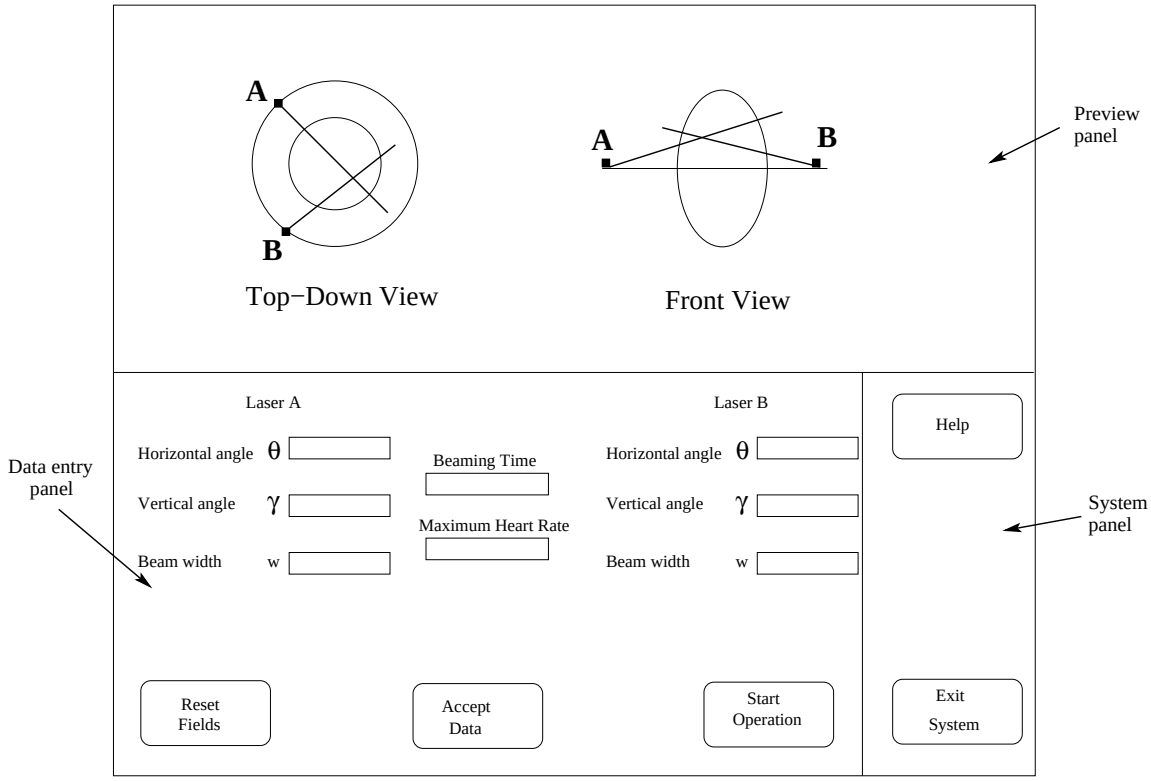

Figure 1: The initial screen of the user interface

- There is a chance that a patient's head may move during the treatment, which could cause injury to the patient, as the treatment area has moved.
- The radiologist must be able to visually monitor the patient during the procedure and monitor their heart rate. There may be circumstances where it would be considered inappropriate to continue treatment.
- Radiologists aim beams that are short in distance because the beam can affect all tissue along its path, even though the beams are strongest at the point of intersection.
- The beams meet at the intersection area, not the intersection point. The beam widths can be adjusted to determine the area of intersection.
- The two lasers must start emitting at the same time and finish at the same time.

The safety considerations for this system are concerned with patient safety alone, as the safety of the radiologist is assured by having them behind a radiation proof barrier.

4.3 Interaction description

The design and operation of a subset of the user interface for the radiation therapy machine is presented in this section. This subset contains the three main interaction screens. The complete description of the interface can be found in Appendix C of [Connelly *et al.*, 2001]. The following section (Section 4.4) details where the patterns were applied and the justification for each. Like all design projects, while performing this design scenario there were many intermediate designs, which for one reason or another were not pursued. Some of these "faulty" designs and justifications for their removal will be provided as supporting arguments during discussion of pattern usage.

Input to the system (regardless of position within the interaction) is via a combination of mouse, keyboard and touch-screen interface. The fields must be filled in with the keyboard, whereas the buttons and entry fields may be selected using either the mouse or the touch screen. A comprehensive session log is maintained by the system, for liability purposes and also to ensure that there is a means of reviewing system information.

The initial screen of the user interface is shown in Figure 1. For ease of description the various interaction areas of the interface have been labelled. The three areas are:

1. the preview panel,
2. the data entry panel, and
3. the system panel.

The preview panel contains an abstract representation of the beam's position within the patient's head. This panel is included to provide a coarse method for the user to check the input data. Large discrepancies

in the input data would be obvious and the radiologist will be able to investigate these discrepancies further.

All data for the treatment is entered into the data entry panel. Any interaction that occurs within this panel affects the treatment. Additionally this panel contains all of the functions related to treating the patient.

The miscellaneous functions that deal with overall system operation are located in the system panel.

Before treatment may commence the radiologist must enter the data into every field within the data entry panel. Once the data is entered into the fields, the radiologist can press the *accept data* or *reset fields* button. If the fields are reset, the preview panel at the top will remain clear, if a previous preview has been performed the system will clear the preview panel. If the *accept* button is pressed the preview panel will update, showing the mapping of the lasers with the given values. The *accept* button cannot be pressed unless there is data in some of the fields. The radiologist may now press the *start* button, the *reset* button, or edit some or all of the fields. If the radiologist attempts to press *start* without first accepting any changes made since the last accept event, or if they attempt to start with empty input fields, an error dialog, accompanied by a error sound, will be shown.

Once the *start* button has been successfully pressed (i.e., there are no erroneous conditions), the radiologist is shown the confirm screen given in Figure 2. In this window all of the fields must be re-entered to ensure the data has been entered correctly into the initial interface. If the radiologist presses the *accept* button, and all fields are accurate, the machine will begin beaming and present the beaming interface screen to the radiologist (shown in Figure 3).

The beaming screen contains two information panels and the emergency panel. The top panel, labelled monitoring panel, contains all of the information that was previously supplied by the interface, as well as the monitoring information of which a radiologist must be aware. The emergency panel contains a single button; this is the *emergency stop* button. This button is not obscured by any error windows presented during machine's operation, and pressing this button at any time will instantly cease beaming, regardless of what mode the system is in. During operation the radiologist will be unable to change the entry fields contained in the data panel, as such the panel will be "grayed out" to show the radiologist that the fields are unmodifiable.

If the information entered in the confirm screen differs from that entered on the initial interface, the system will display the initial screen of the user interface again with all fields cleared, and an error dialog.

During operation a heart rate sensor will be constantly sampling the patient's heart rate. If the heart rate is beginning to approach the preset maximum, the user will be shown an error dialog informing them that the patient's maximum heart rate is being approached. This window will dismiss itself after twenty seconds. Any event that occurs while this window is present will take precedence, that is, this is the only window that does not mode the system. The window will be accompanied by an alarm that will be distinct and will inform the radiologist that the system requires their attention. If the radiologist does not see the screen during the 20 seconds then the alarm will inform them that the patient's heart rate is approaching dangerous levels. The system will continue to alert the radiologist at a random interval that the heart rate is near the maximum by re-issuing the error window, until the patient's heart rate drops, exceeds the defined maximum or the system completes beaming. The random interval must fall within a range that is large enough to minimise sensory overload, yet short enough to maintain awareness of the patient's high heart rate.

If an emergency stop occurs during the system's operation, a window informing the radiologist will be displayed. This window (regardless of type of stop) will contain the time elapsed and time remaining at the instance of stopping. To ensure that there is a record of how much radiation a patient has received and if there is a need to resume the treatment the radiologist must record the timing information before dismissing the window. The initial user interface screen will then be displayed with all fields empty. If the beaming operation completes successfully the initial screen will be re-displayed with empty fields, ready for the next treatment.

4.4 Pattern usage

This section discusses the patterns that were applied in the synthesis of the user interface described in the previous section (Section 4.3). The discussions here will also serve as a justification for each patterns application.

One of the fundamental patterns that was applied to this user interface is the Transaction pattern. All of the information is entered into the system prior to the treatment. The radiologist must then commit this information before the treatment can begin thus allowing all of the data to be checked and verified before beaming is initiated. Combined with this Transaction system is a powerful Task Conjunction. The confirmation window is included so that there can be no confusion of the intended parameters. The radiologist should already have checked the validity of the parameters via the Preview panel at the top of the initial interface. However, this alone was not strong enough due to the lack of detail in the Preview system. Providing an exact Reality Mapping of the patient's head for the radiologist to examine would have been extremely difficult (how would the image be projected into the system?) and unnecessary. To provide the radiologist with an Abstract Mapping makes more sense, as it aids the radiologist in visualising the laser's po-

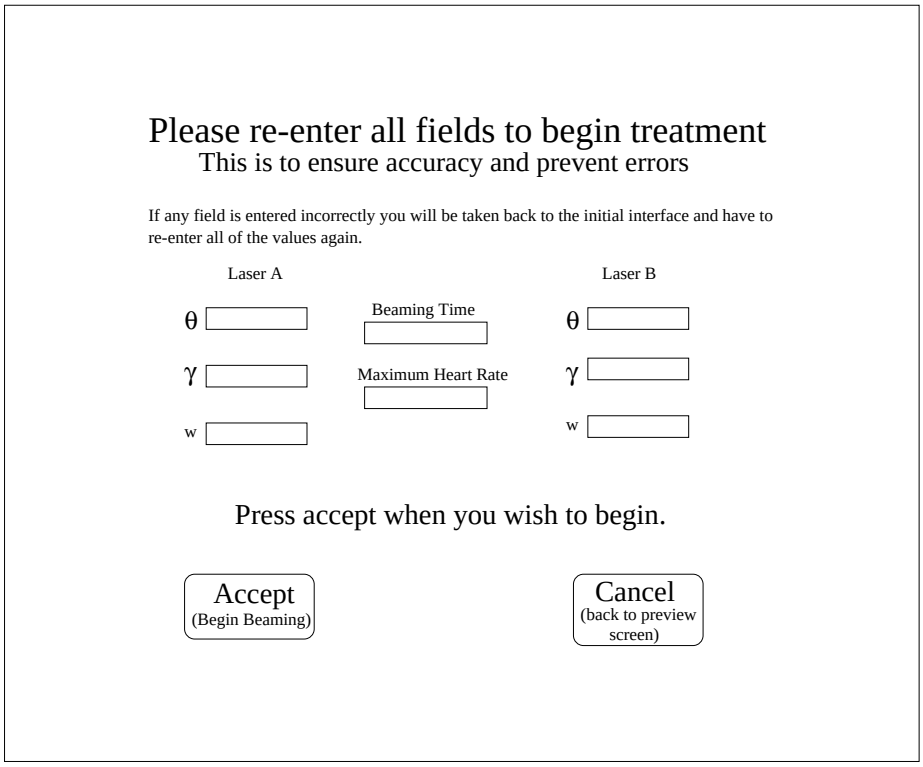

Figure 2: The confirm screen of the user interface

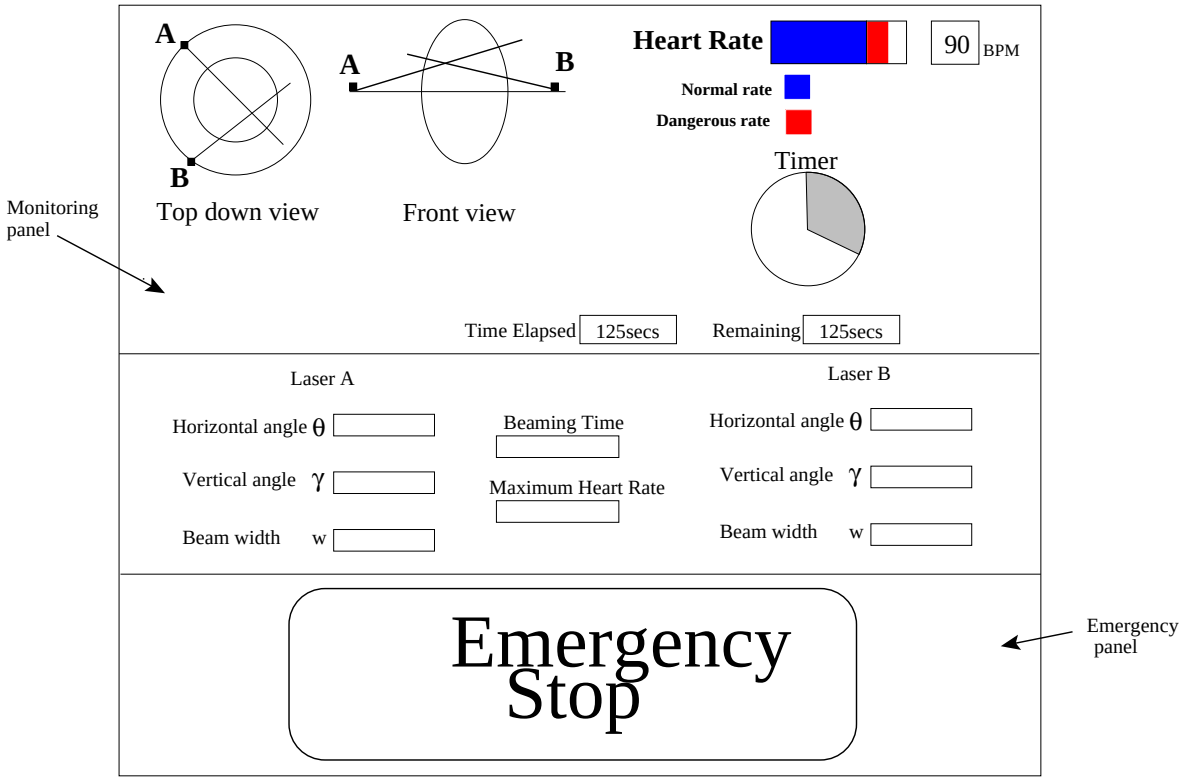

Figure 3: The beaming screen of the user interface

sition within the patient's head. Enabling the radiologist to view the parameters of both beams on the initial interface represents a form of Memory Aid. By allowing the radiologist to view all input fields on the one screen, the design ensures that any empty fields should be obvious to the radiologist.

The buttons within the interface have been laid out in such a manner that they are separated (the Separation pattern) from each other by enough screen real estate that they are unlikely to be accidentally activated. In the initial interface designs all of the action buttons were located very close to each other; this increases the chance of a "mis-click" and hence incorrect operation of the system.

The design of the emergency stop button Affords pressing, as it resembles its real world counterpart. Indeed the button may be physically pressed through the use of the touch screen. It is important that this button be easily identifiable and easy to operate as in the event of an emergency the radiologist must act quickly to prevent damage to the patient.

The inclusion of the Preview panel to the initial and beaming interfaces follows the Redundant Information pattern, as it is a redundant view of the parameter fields. Additionally, providing a numeric and graphical representation of the time remaining represents another application of the Redundant Information pattern.

The dialog boxes that are raised when a radiologist performs an error are an application of Notification. These notifications are multi-media signals as they are combined with sound events. The two types of Notification implemented in the interface are the critical stop and system event notifications.

Within the beaming interface the Alert pattern has been applied to the emergency stop and heart rate warning events. The application of this pattern helps to ensure that a radiologist will react to the Alert situation.

Behaviour Control has been used in the beaming interface via greying out the input fields during operation, this pattern has also been applied anywhere within the interface that "grays put" actions. Behaviour Control was chosen over the Affordance pattern since the introduction of moding into the system was justified. The addition of moding is justified, since there will never be a situation where a radiologist should be allowed to modify the fields whilst the machine is beaming.

The Interlock pattern is applied to the motion sensing part of the system. If the patient is detected to have moved significantly, beaming is halted immediately by powering down the beams. Additionally an error window is raised that contains a record of the time elapsed and time remaining when power down occurred. The error window also requires the radiologist to record the elapsed and remaining beaming time, implementing the Training for Uncontrolled Environments pattern. The training pattern was applied to provide a physical record of the final system state.

In the Abstract Mapping of the heart rate the colours used are blue for normal and red for abnormal heart rate. The choice of these colours resulted from the application of the Accessibility pattern (as discussed in Section 3) and was used to lessen the possibility of problems associated with red-green colour-blindness.

It could be argued that the Stepladder and Interrogation patterns are also implemented in this user interface. The interaction of entering fields, pressing accept and finally pressing start are in the form of a Stepladder. This is because the actions must be performed sequentially. However, the guidance aspect of the pattern is not strongly represented. As the system maintains a log of error messages and session information it could be said that a weak form of Interrogation has been used. The radiologist can go through the session log and obtain information about previous runs of the system, but this is only a very simple Interrogation.

4.5 Discussion

This section has presented a detailed outline of a user interface for a radiology therapy machine specified by Mahemoff. This case study provides a qualitative validation of our additions to the pattern language. By making the interaction flow smoothly and simply, with few complex forks, the usability principles have also been satisfied.

5 Summary and conclusions

This paper has provided mechanisms for capturing the design characteristics of safety-critical user interfaces that reduce operator error. These design characteristics have been expressed as patterns. The aim of providing a design pattern language is not to provide something that a designer must use. Pattern languages are intended as a tool that designers can use as they see fit and change if they feel they have a better solution.

Although we have made major additions to the initial pattern language and modified the categorical placement of the patterns, we do not consider the language to be complete. In our opinion a pattern language can never be considered to be complete because the state-of-the-art keeps changing in the software world and designers are encouraged to modify the patterns within a language. Additionally, if a designer identifies a new problem and outlines a solution, it can become a part of the language.

In Section 3, we presented in-depth information about our modifications and additions to Hussey's pattern language. The new patterns were found by generating problem definitions from perceived exemplary solutions. This approach is a reversal of the usual intervention process applied as part of a Human Re-

liability Assessment. We outlined the pattern structure to provide a standard layout for ease of pattern identification and understanding. The pattern structure used in this document is the same as that used in the previous literature, since maintaining a standard layout aids in the integration of a pattern from one language into another. This structure has been explained in-depth so that future designers will have little trouble synthesising their own patterns.

In Section 4, a test of the pattern language was performed by synthesising a user interface for a hypothetical radiation therapy machine. The case study provided a qualitative check of the validity of our additions to the initial pattern language, and demonstrated that a system designed with our additions to the pattern language would satisfy the basic usability principles outlined in Section 2.

Possible future work would include performing a safety analysis of the user interface developed to access the residual risk associated with the user interface and to determine its tolerability.

In this paper we have identified many problems with safety-critical user interfaces. Throughout this paper we have provided methods enabling designers to increase the safety of user interfaces by controlling the hazardous effect of utilising human operators. We believe that a pattern-based approach to design is extremely useful, and if the language were extended to encompass the overall design of safety-critical systems, a greater number of safe systems would result.

References

[Alexander *et al.*, 1977] C. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language*. Oxford University Press, 1977.

[Alexander, 1979] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[Apple Computer, 1987] Apple Computer. *Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley, 1987.

[Bainbridge, 1987] L. Bainbridge. Ironies of automation. In J. Rasmussen, K. Duncan, and J. Leplat, editors, *New Technology and Human Error*, chapter 24, pages 271–283. John Wiley and Sons Ltd., 1987.

[Commonwealth of Australia, 1998] Commonwealth of Australia. Australian Defence Standard DEF(AUST) 5679: The Procurement of Computer-based Safety Critical Systems. Dept. of Defence, 1998.

[Connelly *et al.*, 2001] S. Connelly, J. Burmeister, A. MacDonald, and A. Hussey. Extending and evaluating a pattern language for safety-critical user interfaces. Technical Report 01-01, Software Verification Research Centre, University of Queensland, Brisbane 4072, Australia, January 2001. http://svrc.it.uq.edu.au/Bibliography/svrc-tr.html?01-01.

[Gamma *et al.*, 1994] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, MA, 1994.

[Hussey and Atchison, 2000] A. Hussey and B. Atchison. Safe architectural design principles. Technical Report 00-19, Software Verification Research Centre, The University of Queensland, Brisbane 4072, Australia, July 2000. http://svrc.it.uq.edu.au/Bibliography/svrc-tr.html?00-19.

[Hussey and Mahemoff, 1999] A. Hussey and M. Mahemoff. Safety Critical Usability: Pattern-based Reuse of Successful Design Concepts. In M. McNicol, editor, *4th Australian Workshop on Safety Critical Systems and Software*, pages 19–34. ACS, 1999.

[IBM, 1992] IBM. *Object-Oriented Interface Design: IBM Common User Access Guidelines*. Que, Carmel, Ind., 1992.

[International Electrotechnical Commission, 1997] International Electrotechnical Commission. 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. IEC, 1997.

[Kirwan, 1990] B. Kirwan. Human reliability assessment. In *Evaluation of Human Work*, chapter 28. Taylor and Francis, 1990.

[Leveson, 1995] N. G. Leveson. *Safeware, system safety and computers*. Addison-Wesley, 1995.

[Mahemoff and Johnston, 1998] M. J. Mahemoff and L. J. Johnston. Principles for a Usability-Oriented Pattern Language. In P. Calder and B. Thomas, editors, *OZCHI'98*, pages 132–139. IEEE Computer Society, 1998.

[Mahemoff, 2000] M. Mahemoff. Software usability study(course notes). Department of Computer Science and Software Engineering, The University of Melbourne, 2000.

[Microsoft, 1995] Microsoft. *The Windows Interface guidelines for Software Design*. Microsoft Press, 1995.

[Open Software Foundation, 1993] Open Software Foundation. *OSF/Motif Style Guide: Revision 1.2*. Prentice Hall, 1993.

[Sun Microsystems, 1989] Sun Microsystems. *Open Look Graphical User Interface Application Style Guidelines*. Addison-Wesley, 1989.

A The initial pattern language

The patterns described in this appendix are those patterns from Hussey's pattern language that were not significantly modified. They have been included for completeness. The full text of the modified pattern language may be found as Appendix A of [Connelly *et al.*, 2001]. The unmodified pattern language can be found in Appendix A of [Hussey and Mahemoff, 1999].

A.1 Task management

Recover If a system enables users to place the system in a hazard state, then there must be a facility to recover.

Stepladder Systems that require operators to perform complex tasks split the tasks into a hierarchy of discreet simple tasks.

Task conjunction To enable the system to check for consistency. The system can check that a user's action matches their intention by requiring that they repeat tasks.

Transaction The need for recovery facilities in a system is lessened if related steps are bundled into transactions, so that the effect of each step is not realised until all the task steps are completed and the user commits the transaction.

A.2 Task execution

Affordance User interface components should be designed in a manner that indicates their operation.

Separation Components of the user interface that are operated in a similar fashion, and for which incorrect operation could be hazardous, should be physically or logically separated.

Distinct interaction Components of an interface that could be confused if they were to be operated in a similar fashion should be operated by distinct physical interactions.

Preview Where the consequences of an action could be undesirable, the user should be able to obtain a preview of the outcome of the action.

Behaviour constraint Users should be prevented from requesting hazardous actions by anticipating such actions and allowing only those that are safe.

A.3 Information

Reality mapping There should be a close representation of the system's components to facilitate user understanding of the system state.

Abstract mapping Where reality mapping is too complex or difficult to provide, the system should provide an abstract mapping of reality to aid in a user's awareness of system state.

Redundant information Where information presented to the user is complex or could be misinterpreted, the system should provide multiple views so that the likelihood of error is reduced.

Trend Humans are not good at monitoring, so when the system state changes, the system should compare and contrast the current state with previous states.

Interrogation Presenting the entire system state might confuse a user. To prevent this confusion, only information that is immediately relevant to the system state should be presented. To provide completeness the user should be able to query the system for more information, as required.

Memory aid If users have to perform interleaved tasks, mechanisms should be provided for recording information about the completion status of tasks.

A.4 Machine Control

Interlock Interlocks provide a hardware-based way of detecting and blocking hazards, so that even if an error occurs, its consequences will be averted.

Automation If a task is unsuitable for human performance, it should be automated.

Shutdown The system should shut down immediately in the event of a hazard situation when shutting down a system is simple and leads to a safe state.

Warning Warning devices should be provided that are triggered by the state to ensure users will notice hazardous system states when they have arisen.