SOFTWARE VERIFICATION RESEARCH CENTRE

THE UNIVERSITY OF QUEENSLAND

Queensland 4072
Australia

TECHNICAL REPORT

No. 01-01

Extending and Evaluating a Pattern Language for Safety-Critical User Interfaces

Simon Connelly Jay Burmeister
Anthony MacDonald[†][*]
Andrew Hussey

June 2001

Phone: +61 7 3365 1003
Fax: +61 7 3365 1533
http://svrc.it.uq.edu.au

Also published in *Proceedings 6th Australian Workshop on Safety Critical Systems and Software*, Brisbane, July 2001. Australian Computer Society.

Extending and Evaluating a Pattern Language for Safety-Critical User Interfaces

Simon Connelly* Jay Burmeister† Anthony MacDonald†* Andrew Hussey ‡

Abstract

This paper describes the extension and evaluation of an existing pattern language for safety-critical user interface development. 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 [16] 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 [2, 8] 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) [11], 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) [14] 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

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

†School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Qld 4072, Australia. email: {jay, anti}@itee.uq.edu.au

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

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 [11] 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 [23] 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 [3, 12, 20, 26] 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 [18] 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" [2]. Alexander proposed using his patterns [1] 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" [8]. Mahemoff and Johnston [18] 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 [11].

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 [11] 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 [10] discuss user-interface principles that are found in several major safety standards [6, 13]. 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.

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 [14]) 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.

[1]The chosen example interfaces are presented in Appendix B

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 categorical placement was a new method of presenting the language. An index to the patterns in the language is provided.The patterns are listed alphabetically in their respective category/categories[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 resulting from modifications of existing patterns are described in Section 3.3.2. A description of all the 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 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 [4].

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.

[2]As shown in Appendix A

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 [17]. 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.
- 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.

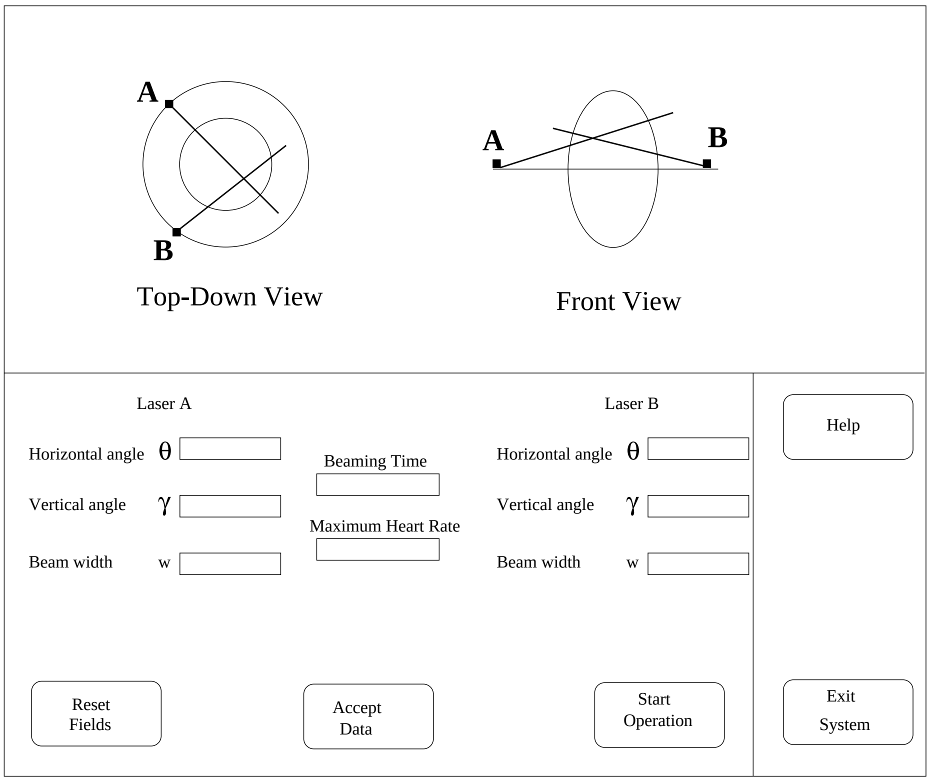


Figure 1: The initial screen of the user interface

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. 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.

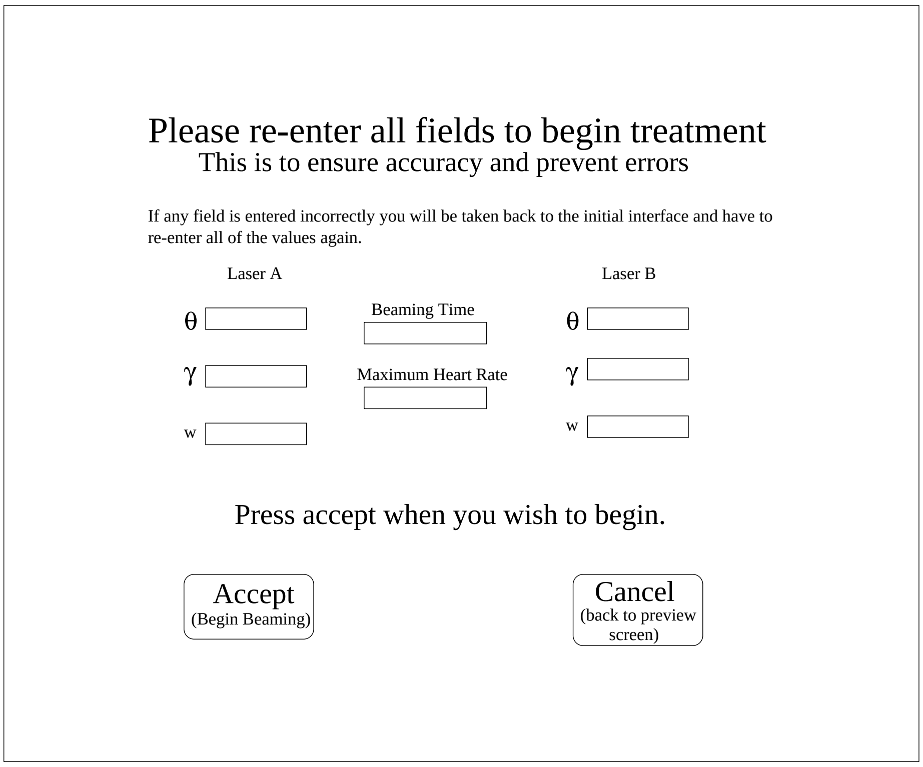
If an emergency stop occurs during the system's operation, a window informing the radiologist will be displayed. This window (regardless of type of stop) will contain the time elapsed and time remaining at the instance of stopping. To ensure that there is a record of how much radiation a patient has 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 position 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



Please re-enter all fields to begin treatment

This is to ensure accuracy and prevent errors

If any field is entered incorrectly you will be taken back to the initial interface and have to re-enter all of the values again.

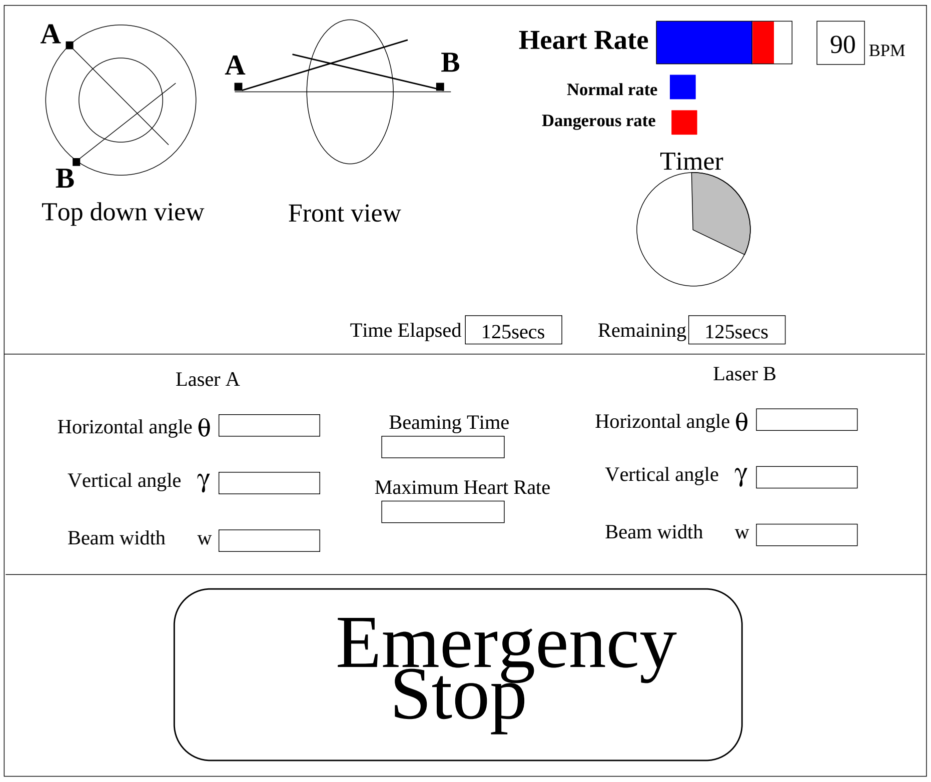
Laser A			Laser B	
θ		Beaming Time	θ	
γ		Maximum Heart Rate	γ	
w			w	

Press accept when you wish to begin.

Accept (Begin Beaming)

Cancel (back to preview screen)

Figure 2: The confirm screen of the user interface



Time Elapsed 125secs

Remaining 125secs

Laser A			Laser B	
Horizontal angle θ		Beaming Time	Horizontal angle θ	
Vertical angle γ		Maximum Heart Rate	Vertical angle γ	
Beam width w			Beam width w	

Emergency Stop

Figure 3: The beaming screen of the user interface

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 Reliability 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

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

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

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

[4] 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.

[5] L. Bainbridge and S. Antonio Ruiz Quintanilla, editors. *Developing Skills with Information Technology.* John Wiley and Sons Ltd., 1989.

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

[7] A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction.* Prentice Hall, 1998.

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

[9] K. Hinckley, R. Pausch, J. Goble, and N. Kassel. A three-dimensional user interface for neurosurgical visualization, 1995.

[10] 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.

[11] 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.

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

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

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

[15] T. Kletz. Plant design for safety: a user-friendly approach. *Hemisphere*, 1991.

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

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

[18] 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.

[19] J. McDermid and T. Kelly. *Industrial Press: Safety Case.* High Integrity Systems Engineering Group, University of York, 1996.

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

[21] D. Norman. The 'problem' with automation: inappropriate feedback and interaction, not 'over-automation'. *Philosphical Transactions of the Royal Society of London, Series B*, 327(1241):585–593, 1990.

[22] D. A. Norman. *The Psychology of Everyday Things.* Basic Books, 1988.

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

[24] R. D. Patterson. Auditory warning sounds in the work environment. *Philosphical Transactions of the Royal Society of London, Series B*, 327(1241):485–492, 1990.

[25] Physio Control Corporation. *Lifepack 500 Information Booklet.* Physio Control Corporation, 1999.

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

A The modified pattern language

This appendix contains the entire pattern language including the modified patterns from Section 3.3. It extends the pattern language found in Appendix A of [11].

The patterns are partitioned into 4 categories: *Task management*, *Task execution*,*Information*, and *Machine control.* Patterns may appear in several categories as presented in Table 1. The full patterns are then presented in alphabetical order. The patterns contain medical examples and these are fully described in Appendix B.

Task Management	**Task Execution**
Accessibility (A.2)	Affordance (A.3)
Recover (A.16)	Behavioural Constraint (A.7)
Stepladder (A.20)	Distinct Interaction (A.8)
Task Conjunction (A.21)	Interrogation (A.10)
Training for uncontrolled environments (A.22)	Memory Aid (A.11)
Training for uncontrolled interfaces (A.23)	Preview (A.13)
Transaction (A.24)	Proximity (A.14)
	Separation (A.18)
	Trend (A.25)
Information	**Machine Control**
Abstract Mapping (A.1)	Alert (A.4)
Alert (A.4)	Automation (A.5)
Automation Feedback (A.6)	Automation Feedback (A.6)
Notification (A.12)	Interlock (A.9)
Reality Mapping (A.15)	Shutdown (A.19)
Redundant Information (A.17)	

Table 1: Pattern Categories

A.1 Abstract Mapping

Context
The system enables users to interact with complex real-world objects.

Problem
Modern computer-based systems may have an extremely complex internal state. To operate the system, the user needs to be able to understand the current state of the system. **How can we reveal to the user what they need to know about the state of the system without overloading them with information?**

Forces

- Humans have a limited capacity for processing information.
- For many safety-critical systems, the amount of information that is relevant to the user's operation of the system exceeds the processing capacity of the user.
- Many decisions that the user must make in ensuring safe operation of the system depend on the overall trend of the system state, rather than precise details.
- Many systems allow some margin for error so that precise representation of the system state is not necessary for safe operation.

Solution
Provide an abstract representation of complex information so that it is comprehensible by the user. Abstract representations can be used when it is not helpful to directly represent the real-world objects. One benefit of computers is that we can construct summaries or indirect mappings which aid human decision-making. Users should not have to perform complicated mental operations to assess the margins of system operation. More abstract representations can be used in situations where we are prevented from showing real-world objects. For instance, bandwidth might be constrained. Displaying unnecessary parameters increases system load but redundant parameters can be used to automatically check the consistency of information, which can then be displayed as a single value.

Examples
The Druide air-traffic control interface outlined by Hussey [11] displays the speed of each plane as an abstract mapping. Speed is a variable which could be derived from the reality mapping, but is shown explicitly to the user, because a user's estimate would lack accuracy, consume time, and distract them from other tasks. The Defibrillator system summarises the state of the patient as either suitable for shock or unsuitable for shock; an extreme form of abstract mapping. Displays of pulse amplitude against time are abstract mappings of the patient's pulse strength.

Resulting Context
Information will be displayed to the operator using abstract representations unless a closer Reality Mapping is feasible.

A.2 Accessibility

Context
Users of safety critical systems cannot be guaranteed to have the same abilities or cultural grounding as those for whom the system was designed.

Problem
How can we ensure that mistakes and misrepresentation of information does not occur due to user misunderstanding?

Forces

- Users of safety-critical systems are assumed to have no major disabilities.
- All users of a system cannot be assured to be completely devoid of certain (non-obvious) disabilities.

- Users of a safety-critical system will not all be from the same cultural background as the designers of a system.
- There may be vast cultural differences within the user population.

Solution
Interfaces should be designed to include methods of mitigating misinterpretation of information. The need to cover extreme disabilities is removed due to the fact that there is more control over the users of a safety-critical system. It is important to realise that this solution is merely a placeholder for a larger sub-set of patterns for safety-critical systems.

Examples
No examples are provided for this pattern because this pattern covers a much larger subset and therefore no example would be indicative of a solution to the entire problem set.

Resulting Context
Once this pattern has been applied there should be a reduced likelihood of misinterpretation of signals by the user. It is important to realise that this is an incomplete pattern. The problem covered in this pattern is more of a defining context for an entire subset of patterns.

A.3 Affordance

Context
A task has a limited range of valid ways in which it can be performed and failure to perform the task correctly has hazardous consequences. For many systems, it is possible for the user to perform a variety of physical actions at each point in performing the task; not all will produce the required actions for the task.

Problem
How can we enhance assurance that the physical actions performed by users produce the desired effect?

Forces

- It is possible for the user to have the right intentions, but perform the wrong action due to a slip.
- Slips can be reduced by providing appropriate affordances [22].

Solution
Provide cues to an operator that enhance the likelihood that the operator will perform the physical actions appropriate to performing a particular action. The cues are effectively memory aids that remind the user to perform an action by a particular sequence of executions, avoiding slips. Such cues include distinctive executions for distinct actions and distinctive identifiers for distinct objects that the user can manipulate. The actions performed are matched to the outcomes of the actions and the user's physical expectations and capabilities. In addition, the same execution should not have several different outcomes according to the model or type of equipment that is in use. Incorrect executions should clearly indicate that the operation has not been performed successfully.

Examples
Norman [22] gives several examples of doors that provide affordances. The physical characteristics of the door indicate the way in which it should be used. For example, a door knob may indicate by its shape that it should be twisted, whereas a flat bar on a door indicates that the door should simply be pushed. Failure to perform the operation of opening the door is indicated clearly because the door remains shut. Similarly, the landing gear on an aircraft is operated by a lever that is pulled downwards, mimicking the effect of the operation on the position of the aircraft's undercarriage. Graphical toolkit components such as tabbed dialogues afford correct action by the user, because they mimic "real-world" entities that offer particular operations and indicate the availability of those operations by physical cues.

The placement of the buttons on the Defibrillator is indicative of how the system should be used. As the system is designed to be used by "Westerners" (who read from left to right) the left to right placement suggests how the system should be used.

Design Issues
Selection of cues depends on the user's perceptual limitations. A user may not notice small differences in shape if they are occupied by other activities. General user characteristics must also be considered, e.g., distinguishing between red and green might not be appropriate if colour-blind users interact with the system. Customisation of the interface may be necessary to accommodate the needs of all users in the local organisation. This customisation should follow the guidelines put forth in the Accessibility pattern.

Resulting Context
Certain types of user errors ("slips" as described by Norman [22]) have reduced likelihood. Notification should be used to notify the user if an operation has not succeeded. Alet should be applied if such a failure leads to an identifiable hazard state. There should be a Reality Mapping to provide the user with a representation of the state of the system, so they can determine whether the operation has occurred correctly. Because errors may occur, Recover should be applied to enable the effects of error to be recovered from where possible. For those errors for which risk is too high for this pattern to be applied, Behaviour Constraint or Interlock should be considered. If the Affordance involves the use of toolkit components, Separation should be considered. A Preview is a simple way of affording correct actions by showing what the outcome of an action will be.

A.4 Alert

Context
There are identifiable hazard states, that indicate whether an error or fault has occurred.

Problem
How can we ensure that the user responds to any hazard states entered into by the system that cannot be handled via machine control?

Forces

- Safety-critical systems generally have a very complex system state.
- Computers are good at monitoring state.
- The most common human error in a system hazard is misdiagnosis.
- Auditory signals of an appropriate range, within the existing noise level, can aid in getting the users attention.
- There should be only 4-6 distinct alarms within a system as this is the limit of easily remembered alarms [24].
- Users will have a correct mental model of current system state.
- During long running and repetitive processes, users will not always maintain their full attention on the system.
- Users are highly trained.
- In an automated system the logic may handle some errors by itself without user intervention.
- Moding a system must not interfere with important actions.

Solution
Provide appropriate alarms for system hazards. These alarms should provide information about the error such as type, location, cause etc. The alerts provided by the system should be brief and simple. Spurious alarms should be minimised, reducing the total number of alarms to a minimum. Users should have access to any further information that may affect their decision as to the cause of the problem. The form of an alarm should indicate the degree of urgency and which condition is responsible. When an alarm arises, the user should be aware of the system state due to adequate Notifications within the system, since such awareness will lessen the chance of misdiagnosis. The user should be guided to the disturbed part of the system and aided in the location of disturbed parameters in the affected system area. Alarms should be provided well in advance

of the point at which a serious accident is likely. The absence of an alarm should not be actively presented to users. Alarms should be regarded as supplementary information; a system should never be designed to operate by answering alarms, that is, a system should not be designed to be run in a reactionary manner.

Examples
The Handling Alarms Using Logic (HALO) system (discussed by Bainbridge [5]) uses logical expressions to generate a reduced number of alarms from a total array of alarm signals. The HALO system aims to provide meaningful alarms to users that provide more information than just "the system is broken". HALO alarms inform the user of what the problem is and where it is located through an alarm display that is incorporated into the system. The system provides a "filter" over all alarms, preventing the user being assailed by multiple alarms and responding in a panic [21]. This filtering lowers the number of total alerts that a system may raise, hence reducing confusion (as shown by Patterson [24]). The panic response is to switch off all alarms and then try to ascertain what has gone wrong. Unfortunately in such a situation the only indicators of the problem have been removed (i.e., all of the alarms have been turned off).

Design Issues
It has been found that only 4-6 alarms should be used within an alarm set. It is also important to take into account the modulation of an alarm. When choosing the tone and modulation of an alarm it is necessary that it cannot be confused with other alarms such confusion could be caused by an alarm sounding at the same time as another alarm, and the two alarms forming a completely different sound. Another consideration is the broadcast nature of sound. This means that not only does the designer need to take into account the alarms attached to the system being designed, but also the alarms of surrounding systems.

Resulting Context
For this pattern to be effective it is important that the user be constantly aware of the internal state of the system. Such awareness may be achieved through correct usage of the Notification pattern. Alerts should be replaced by Automation and the associated Notification where possible. The main operational difference between a Notification and an Alert is that a system can handle events related to Notifications where an Alert relates to a system state unrecoverable through any means other than user interaction. Alerts may be structured hierarchically, so that only primary failures that are responsible for a system or sub-system failure are displayed initially, with secondary failures appearing only at the user's request (i.e., applying the Interrogation pattern). If an alert is triggered, the user should have access to mechanisms for recovery (Recover) where possible.

A.5 Automation

Context
Consider this pattern if performing a function involves danger to a user, performing the function requires tedious or repetitive work, or performing the function requires exceptional skill, e.g., as when the response time is far shorter than a human can normally achieve.

Problem
Many safety-critical processes, such as nuclear power generation and advanced medical treatments such as radiation therapy, require manipulation of a large number of parameters to keep the system within safe margins of operation. However, humans are not very good at monitoring and controlling a large number of parameters in real-time. Machines are good at such monitoring and control but typically cannot detect and correct all possible abnormal component failures. **How can system parameters be reliably maintained within safety margins even in the presence of component failure?**

Forces

- Part of the operation of the system involves maintaining parameters within defined safety margins.
- Users become fatigued when monitoring parameters to ensure they stay within safety margins.
- Users need to remain informed of what the system is doing so they can intervene in the event of component failure.

Bainbridge [4] maintains that it is not possible for even a highly motivated user to maintain attention toward a source of information on which little happens for more than half an hour. Hence it is humanly impossible to carry out the basic monitoring function needed to detect unlikely abnormalities. In addition, the user will rarely be able to check, in real-time, the decisions made by the computer, instead relying on a meta-level analysis of the "acceptability" of the computer's actions. Such monitoring for abnormalities must therefore be done by the system itself and abnormalities brought to the user's attention via alarms.

Solution
Automate tasks which are either too difficult or too tedious for the user to perform. Hussey [11] suggests that a function should be automated if:

- performing the function involves danger to a user;
- performing the function requires exceptional skill, e.g., as when the response time is far shorter than a human can normally achieve; or
- performing the function requires tedious or repetitive work.

It should not be fully automated (i.e., a human should be included in the control loop with responsibility for decisions) if a decision must be made that:

- cannot be reduced to uncomplicated algorithms;
- involves fuzzy logic or qualitative evaluation; or
- requires shape or pattern recognition.

Appropriate design of automatic systems should assume the existence of error, it should continually provide feedback, it should continually interact with users in an effective manner and it should allow for the worst situations possible [21].

Examples
Typical examples of automation are cruise control in cars and autopilots. Autopilots reduce pilot workload; without autopilots, pilot fatigue would be a very significant hazard in flight. Further, some aircraft (e.g., military aircraft) could not be flown without automatic control of some aircraft functions. However experience from case studies of crashes indicates that there is not always enough feedback and that this can lead to accidents [21]. The Defibrillator automates the analysis of heart rate, blood pressure and the consequent decision as to whether a shock is advised or not.

Design Issues
The automated system needs to provide continuous feedback (Automation Feedback) on its behaviour in a non-obtrusive manner. Careful consideration is therefore required of which aspects of the controlled process are most salient to safety and to ensure prominence of their display. If the system has been given sufficient intelligence to determine that it is having difficulty controlling the process, then active seeking of user intervention is appropriate.

Resulting Context
Aspects of the system that are unsuited to human performance are automated. Automation of a system will usually require that the system also notify (Notification) the user of failures that have been handled by the Automation. If a failure is unable to be handled by the Automation then an Alert should be raised.

A.6 Automation Feedback

Context
The concept of abnormal state is not one that is easy for automation logic to emulate. When an error occurs the system will begin correcting for the error as if were a normal system state. An automated system produced with current technology will not be able to completely handle all errors.

Problem
When an automated system attempts to correct for errors with unrecoverable limits it may actually worsen the problem. **How do we ensure that automatic systems, through their dealing with system errors, do not exacerbate the existing error condition?**

Forces

- It is difficult to define all possible points of failure within a system.
- It is difficult to specify all of the behaviours of a system considered abnormal.
- Users cannot be expected to maintain full attention on an automated system [4].
- Existing artifical intelligence (AI) for automation is immature in its ability to handle non-standard system operation [21].

Solution
The solution provided by this pattern is not optimal, as some forces cannot be resolved. In particular the inability to handle non-standard operation of a system is currently un-resolvable, since the problems with automation logic are still a large problem. **At all stages of the operation of an automated system's running, the user should be made aware of the internal state of the system through obvious abstract mappings of trends in the system's operation. This will ensure that the user will be able to rectify a hazard state if it arises.** Providing this information to users for every change of system state will make obvious where the change originated and whether from user input or other parameter changes. To aid viewing the changing system state, the Trend pattern should be applied in combination with the Interrogation pattern. This will allow the user to view any dubious system changes in more detail by querying the system. The reason that Abstract Mapping has been chosen over Reality Mapping is that in complex real-time systems all of the information relating to state needs to be abstracted to make it understandable.

Example
As there were no good examples of the solution provided by the Automation Feedback pattern a hypothetical solution to the aircraft problem will be discussed [21]. The outlined problem deals with the loss of engine power in a China Airlines flight. The only indication provided by the automation to the flight crew of an error within the system was when the flight stick independently began to turn towards full rotation. The Automation needed to provide more descriptive feedback to the flight crew. A possible solution to this problem was to provide the flight crew with a display that shows the current system yaw-state[3]. This display should also contain an indication of the yaw-state change (the Trend). A low level verbal Notification should also be attached to the system to warn the flight crew if the yaw-state reaches a critical level.

Resulting Context
After this pattern is applied the system should provide more information to the user as to the constant state of the system. The main concern is that users do not have too much input or that the system is not too intrusive. Most of the information presented by this pattern will be provided through effective use of the other information patterns (Notification).

A.7 Behavioural Constraint

Context
It is possible to determine system states where certain actions would lead to an error.

Problem
The most direct solution to improving assurance through diminished user error is to prevent users from making errors. **How can we prevent users from requesting undesirable actions?**

Forces

- Even if we provide appropriate information and guidance by providing Affordances, users will inevitably make some slips and mistakes.
- The system has a large amount of data available concerning the state of objects.
- Designers will be aware that certain tasks will be undesirable in certain situations.
- It is preferable to trap user errors before they impact on the system, rather than to detect potentially hazardous system states and then attempting to rectify the situation.

[3]The yaw-state is the degree of turn on the flight deck.

Solution
For any given system state, anticipate erroneous actions and prevent the user from performing them. The idea here is to prevent the action from occurring in the first place, rather than dealing with whatever the user does.
This is a very fragile approach which should be used with extreme caution. It assumes that the condition is measured accurately and that ignoring the constraints always implies a worse hazard than following them. It is therefore risky in unforeseen circumstances. This can be somewhat alleviated by a mechanism allowing an appropriate authority to override the constraint.
A further disadvantage is that this pattern leads to a less flexible user interface (i.e., the interface is moded) and the user may become frustrated with the structured form of interaction if they do not properly understand the tasks they are performing. In a safety-critical domain, this should be a less severe problem than normal, because the user should be highly trained.
Behavioural constraints are usually implemented as an additional system mode, e.g., in a graphical interface "greying out" menu items or buttons where appropriate. As such, behaviour constraints require close automatic monitoring of system state and therefore a frequently used partner pattern is Automation.

Examples
Kletz [15] gives everyday examples, such as components that will only fit together in one way, making incorrect assembly impossible. Norman [22] also gives such everyday examples, such as fire stairs that have a rail blocking access to the basement stairs; the error of proceeding all the way to the basement is prevented.

Resulting Context
Some user errors are no longer possible. The system will usually be more heavily moded than prior to applying the pattern. The system may inform the user that a requested operation is unavailable (i.e., applying the Notification pattern). When interface cues are provided to inform the user that a function is no longer available the Accessibility pattern should be applied.

A.8 Distinct Interaction

Context
Two or more tasks are performed by a similar sequence of physical actions, and confusion between the tasks may result in a hazard. In addition, it is not possible to predict hazards and prevent the potentially hazardous action at the point at which it might be erroneously selected.

Problem
How can we reduce the likelihood that users will confuse similar tasks?

Forces

- Reuse of graphical toolkit components enhances consistency and makes a user interface easier to learn.
- Reuse of such components also increases the consistency of the operations required to perform tasks, so that distinct tasks may be accessed by similar physical executions.
- Tasks that have similar execution sequences are likely to be confused by users.
- Users confuse tasks because of memory failures or distractions.

Solution
Distinct actions that can be confused and lead to hazardous consequences should be accessed by distinct physical interactions. However, distinct physical interactions reduce reuse, making the system harder to learn. Training for uncontrolled interfaces, and Training for uncontrolled environments, and Memory Aids can help overcome errors arising from users not remembering the correct execution sequence to perform a task.

Examples
The Hypodermic Syringe system uses alignment of +/- buttons with the corresponding display digit to reduce motor errors in which a wrong button is pressed. This strategy is also used in the Angiography system.

Resulting Context
Controls that might be confused by the operator are operated by distinct physical interactions. The interactions required to operate a control should be afforded by the control (see Affordance). Separation and Preview are alternative solutions. If a graphical toolkit has been extensively used then Training for uncontrolled environments should be applied.

A.9 Interlock

Context
Risk is sufficiently high that measures to block the occurrence of error do not give sufficient assurance and the bounds of acceptable system outputs can be defined.

Problem
How can we be sure that errors cannot lead to high risk hazards, even if they occur?

Forces

- Measures to diminish user errors are not necessarily sufficient assurance if risk is sufficiently high.
- Behavioural Constraints may not prevent all incorrect commands because systems are too complex to predict all possible states and events.
- Measures to diminish user errors should not necessarily be regarded as sufficient evidence of system safety and additional evidence may be required if risk is sufficiently high.

Solution
Anticipate errors and place interlocks in the system to detect and block the hazards that would otherwise arise. Interlocks can be embodied in hardware or software, preferably both. But there is no point creating an interlock if the system failure causes the interlock itself to work incorrectly.

Examples
Many modern motor cars come equipped with Anti-Lock Braking Systems (ABS). Such systems are Interlocks, screening aberrant driver behaviour. If the driver presses too hard on the brake pedal, the ABS will override the driver's actions and maintain brake pressure at an optimum rate. The Therac-20 and Therac-25 machines are medical equipment designed to administer radiation doses to patients [16]. The Therac-20 machine has a hardware interlock that prevents extreme doses of radiation but the Therac-25 machine does not. The Therac-25 machine was involved in several well-publicised accidents that arose because of an error in the software which failed to correctly update the dose after the user had corrected it on screen. To detect errors, the hazard situation must be formulated in a straight-forward and error-free way. As an example in which this was not so, consider the Warsaw accident in which A320 braking was prevented because braking logic required both wheels on the ground (e.g., see [16]). These issues are considered further in Automation.

In the Defibrillator example interface the user cannot use the shock button unless the system is in such a state that the use of the button is deemed acceptable. The characteristics of a hazardous state (e.g., movement) are fairly well defined in the case of applying an electric shock to someone's chest region. If the system is about to give a shock, has just analysed or is mid-analysis and movement of the patient is detected the system will halt all action and give the displayed messages "motion detected", "stop motion" and a voice prompt. This not only stops the action being performed, it also informs the user as to the event and allows them to recover from the hazardous situation. Another interlock prevents use of the shock button unless the system deems shock advisable. The MRI system also uses an interlock when detecting if the temperature of the patient is exceeding safe levels. The sensors will immediately stop the MRI system from taking any more scans of an area if the temperature exceeds these levels.

Resulting Context
If a system is designed using only interlocks to prevent hazards arising from user error, removal of the interlocks opens the system to the possibility of hazardous operation. Interlocks therefore should always be used with Behavioural Constraint to provide "defence in depth".

A.10 Interrogation

Context
The system is complex, with much information of potential use to the user in performing their work and not all attributes can be displayed at one time.

Problem
Most safety-critical interactive systems display to the user a representation of the system state. For many such systems, the state of the system is complex and cannot be represented in a comprehensible way. For such systems, displaying the entire system state at one time may obscure the most important components of the state and represents a potential source of user error. **How can the user have access to the entire state of the system without being overwhelmed by information?**

Forces

- Some of the information is more salient, or more often necessary, than other components of the information.
- Some of the information is more readily/easily displayed than other components of the information.
- Users have limited attentional capacity.
- Display devices have limited resolution and capacity and can quickly become cluttered with information. Users have difficulty locating specific features on cluttered screens and this is particularly problematic when urgent information or control is desired.
- Designers cannot realistically envisage every possible information requirement.

Solution
Provide ways for the user to request additional information. Thus, not all information needs to be shown at once.

If the user is monitoring automatic systems, provide independent information on the state of the system in case instrumentation fails; instrumentation meant to help users deal with a malfunction should not be able to be disabled by the malfunction itself.

The capability to interrogate should be obvious or intuitive; menus at the top of the screen or window are preferable to pop-up menus or mouse actions performed on the object of interest. The results of the interrogation should be able to be saved and reviewed at a later time.

Examples
Interrogation of the Defibrillator system may take place by the user downloading all of the stored log information into a computer for viewing. If the patient was en route to the hospital and the data was transmitted ahead via a cell phone connection, the hospital would have invaluable information when the patient arrived.

Resulting Context
The mapping of system state to the display is now a mapping of only part of the state; the remainder of the state is hidden. The result is more efficient use of the display.

A.11 Memory Aid

Context
The task being performed by a user enables arbitrary interleaving of actions, with the possibility that an action may be forgotten.

Problem
Some safety-critical systems, such as air traffic control, require the user to perform several tasks concurrently, with interleaving of task actions. In such systems, the potential for actions to be forgotten is much greater than in non-interleaved systems. **How can users reliably perform interleaved tasks?**

Forces

- Users must remember to finish all tasks, including interrupted tasks.

- For some systems, the user must not inadvertently perform a task step more often than is required (for some systems and steps, a hazard may result).

Solution
Provide ways to record the completion status of steps. This will help the user to recommence later without omitting or repeating tasks. Such memory aids may be either components of the computer system itself, or adjuncts to the computer system. Memory aids may be proactive, cuing the user to perform an action at a particular point in time, or when the system reaches a particular state; such memory aids may be warnings.

Examples
Airtraffic control systems (e.g., Druide [11]) often use paper strips to record flight details and the instructions that have been given to aircraft. The paper strips provide context for the controllers, enabling controllers to determine whether an aircraft is currently behaving in accordance with the instructions that have previously been issued and also enabling the last action performed for a particular aircraft to be recorded. However because paper strips are an adjunct to the computerised ATC system, they cannot *actively* cue the user to perform an action at a particular point in time, or when the system reaches a particular state. Hussey et al. [11] describe checklists as a simple memory aid to ensure that all the steps in a safety-critical procedure are completed (e.g., piloting an aircraft).

The Angiography machine uses a simple form of memory aid by supplying default parameters when the template for a body part is selected. The Dye Delivery system also supplies default values. The Angiography machine also uses a 5-minute timer to remind the operator that they have been scanning for that amount of time.

Design Issues
Proactive memory aids may be set by the user or by the system. However system-initiated warnings require that the system be aware that a user task has not been completed. Memory aids should cue the user with an urgency corresponding to that set by the user or in the case of system initiated warnings, with an urgency corresponding to risk. Passive memory aids should be visible to the user at all times, e.g., tags associated with an object in the display.

Resulting Context
The user is provided with active and passive Memory Aids. Passive Memory Aids may require Reality Mapping. Active Memory Aids may notify (Notification) the user that a condition (user or system defined) has been reached, and that the user should take appropriate action. Trend displays are a form of passive Memory Aid.

A.12 Notification

Context
There are system events of which a user must be aware of.

Problem
How can we be sure that all important changes of state are brought to the attention of the user?

Forces

- During long running and repetitive processes users will not always maintain their full attention on the system.
- Auditory signals of an appropriate range, within the existing noise level, can aid in getting users' attention.
- Users are highly trained.
- In an automated system the logic may handle some errors by itself without user intervention.
- By requiring the user to confirm every single notification the system will be heavily moded. A heavily moded system will run very slowly due to the system being required to wait for user input at every state change.
- Having no confirmation of notifications within a system may lead to users ignoring them, resulting in users not having a sufficient mental model of the system.

- Moding a system must not interfere with important actions.

Solution
There are many times within the usage of a system that a user may be required to be aware of the system state. Awareness of the system state is important because it will ensure that a user's mental model of the system's internal state will always be current. **If an event is important enough to require user attention, but does not constitute a hazard case, then provide an auditory signal, with displayed information that mirrors the event signal (a multi-medium signal).** The signal may be combined with an input request if required. It is important to consider the moding of the system when deciding whether to require input. Only those interactions that are deemed exceptional should be moded and the less important, yet still noteworthy, events should be merely a multi-medium signal. These signals will ensure that users are constantly aware of the system state and important interactions are not missed. When a system is moded it is important that this moding does not interfere with important interactions within a system. An example would be ensuring that users could still terminate the operation of a system while there is a dialog box on the screen. A signal that notifies of a previously entered hazard state, handled by machine control, will ensure that users are aware of all problems encountered during the operation of a system. The user should not be constantly notified of correct operation, that is, when a system is operating correctly only important events should be brought to the user's attention. This is to prevent the user "switching off" to the notifications and ignoring them altogether.

Examples
At each step within the operation of the Life-Pak500 the user is guided through the interaction of the system by various voice prompts telling them the outcome of the previous action and how to proceed from here. All voice prompts are mirrored on the system's display. For example, the Defibrillator notifies the user that the patient has moved by changing the displayed message and also by emitting a voice prompt "Stop Motion". This is a dual notification in that it states that an important change of state has occurred. It also notifies the user that a hazard state was entered into (the patient has moved before a shock was administered) and the system has handled this exception.

Resulting Context
It is important that only those interactions and state changes deemed important should be accompanied by a notification. This pattern is designed to complement the Alert and Automation patterns by helping maintain user awareness of a system's operation. This pattern will also aid the application of Stepladder to a system, by making it easy for the system to guide the user. Overuse of this pattern may lead to confusion among users as to what each sound means. If there are too many notifications in a system they may interfere with the existing Alerts in a manner that may cause misdiagnosis. Another consideration is the affect of having more than one warning occurring at the same time which could lead to misunderstanding of the notification. If overused this solution can cause the system to be too heavily moded and increase user frustration with the interface. Due to the user being highly trained, highly repetitive tasks should not notify of each step. With proper use of this pattern users will be able to deal with any Alerts as they arise with a lessened chance of misdiagnosis, as they will be more familiar with the system's internal state.

A.13 Preview

Context
The same physical action has different outcomes according to the system mode. The user cannot be reasonably expected to recall the current mode.

Problem
How can we provide hints as to the outcomes of physical executions within the constraints of graphical toolkits?

Forces

- Graphical toolkits diminish the extent to which affordances can be incorporated into the design of a system.
- Use of toolkits enhance the consistency of a system design and economic viability of a system.
- Affordances provide cues that indicate to the user the likely outcome of physical executions.

Solution
Provide an explicit preview of the outcome of a physical execution for a system mode. Preview only works well when there is only one execution that can be performed and the issue is whether the user should perform the execution or not, rather than what execution they should perform.

Examples
Changing the mouse cursor according to the effect of physical executions for the screen region that the mouse is over. Postage stamp sized pictures of screen shots for viewer software.

Resulting Context
The operator is cued as to the outcome of actions.

A.14 Proximity

Context
Through the use of a system a user may be harmed.

Problem
How can we ensure that the physical location of an operator during system operation does not present an unacceptable level of risk of harm or injury to the operator?

Forces

- Systems contain components capable of severe maiming or death.
- Some tasks require close monitoring by operators.

Solution
Physically separate the user interface from dangerous sections of a system. When considering the separation of users from components it is important to consider whether or not they will still be able to perform their role. In some instances the user will be so integrated into the system that distancing them from this "dangerous" section may increase the danger.

Examples
There are many examples of the proximity pattern in safety-critical systems. In all radiation therapy and medical x-ray machines the operator is positioned behind an impervious shield. This shield is present to protect the user from the danger of over-exposure to radioactive waves.

In the defibrillator the user is physically separated from the direct danger of holding the electrodes onto the patient's chest. Although the user cannot be too far removed from the patient they are still slightly distanced from the dangerous current. It may be suggested that the user could be distanced even further from the electric current, but if they are taken too far from the patient they will not be able to monitor the patient's condition satisfactorily, hence increasing the danger to the patient.

Design Issues
The main problem that can be caused by applying the Proximity pattern is the problem of isolation [21]. Therefore when deciding to apply this pattern the designer must consider the negative effect of moving a user from the system's point of operation. For example, although moving the defibrillator operator 40 metres away from the patient during treatment would increase the safety of the operator, the danger of a shock being given accidentally to a bystander or to a patient who's heart suddenly starts beating again is significantly increased.

Resulting Context
This pattern should be applied in all situations where the use of the system is potentially dangerous for the user. The major consideration is that by taking the operator away from the dangerous portion of a system you may make the system more dangerous, due to the user being unaware of exactly the system state. When the user is required to remain in the area of dangerous components it may increase safety to apply machine control patterns (e.g., Interlock) to the system that will detect danger to the user and prevent components from injuring them.

A.15 Reality Mapping

Context
The system enables users to interact with real-world objects.

Problem
Digital systems usually create a degree of separation between users and physical objects. In some aircraft, for example, the visual feedback is entirely virtual. Benefits notwithstanding, there is a risk that vital information may be unavailable. **How can the user check that objects under their own control, or under the system's control, are aligned with their expectations?**

Forces

- Information concerning the current state of a safety-critical system needs to be displayed clearly and succinctly, so that the user can quickly ascertain whether a hazard has arisen or could arise, and can take appropriate action.
- When parts of the system have been automated, there is a temptation to assume that the user does not need to see the state of certain objects. However, failures do arise, and human intervention is often necessary [22].
- Information about the environment is necessary to help humans monitor the system, and, if necessary, intervene.

Solution
Provide a close mapping to reality where possible and supplement it with abstract representations. To help the user build an accurate model of the domain, it is important where feasible to maintain a close mapping between physical objects and their virtual counterparts. A close mapping to reality will help the user observe and diagnose problems effectively. The mapping should incorporate the necessary information for the user to ensure safe operation.

Examples
The Druide air-traffic control system provides an accurate, directly manipulable, display of the airspace and the aircraft in it. Reality mapping is also used in the Neurosurgical Planning system, in which a three-dimensional view of the patients skull is provided to the surgeon for manipulation and annotation.

Design Issues
When mapping to reality, the appropriate level of detail will be guided by knowledge of the user's characteristics and tasks. Object-orientation provides a good way to reduce semantic distance (distance between the human's model and the real world) because each object in the display for the system represents a corresponding object in the task domain. Analogue displays also reduce semantic distance because they enable direct comparison of magnitudes. Similarly, minimise articulator dissimilarity so that physical actions mimic their meanings.
In situations when the representation is complex, abstract representations can be used to extract from reality any information which is likely to help users in their task.

Resulting Context The result is a mapping from reality into suitable display objects. Since the display will not be optimal for all cases, the Interrogation pattern can be used to help the user refine the information provided. An Abstract Mapping may be used when Reality Mapping of the system state is not necessary for safety, or is infeasible.

A.16 Recover

Context
A task has been designed which could lead to a potentially hazardous state and it is possible to recover to a safe state from that potentially hazardous state. Risk can be sufficiently reduced by providing recovery paths, rather than reducing error likelihood. Prevention is costly or not possible.

Problem
How can we reduce the likelihood of accidents arising from hazardous states?

Forces

- Hazardous states exist for all safety-critical systems; it is often too complex and costly to trap every state by modelling all system states and user tasks.
- Reducing the consequence of error rather than its likelihood can effectively reduce risk.
- When a hazardous state follows a non-hazardous state, it may be possible to return to a non-hazardous state by applying some kind of recovery operation.

Solution
Enable users to recover from hazardous actions they have performed. Recovering a task is usually similar to undoing it and promises to return the system to a state that is essentially identical to the one prior to the incorrect action. In many safety-critical systems this is impossible. For example, incorrect dosage of a patient by a drug delivery system is impossible to undo once it occurs. However, it may be useful to provide a recover operation giving a fast, reliable mechanism to cancel the current operation. Recovering a task reverses as much of the task as is necessary (and is possible) to return the system to a less hazardous state. This function can be assisted by:

1. Helping users to anticipate effects of their actions, so that errors are avoided in the first place;
2. Helping users to notice when they have made an error (provide feedback about actions and the state of the system);
3. Providing time to recover from errors; and
4. Providing feedback once the recovery has taken place. If this solution is not feasible, a more extreme way to deal with unwanted system states is to perform a Shutdown.

Examples
In the Hypodermic Syringe example interface, users can recover from an error by using the +/- button to alter the entered value to a safe value.

Resulting Context
After applying this pattern, it should be possible for users to recover from some of their hazardous actions. The Stepladder pattern facilitates recovery (Recover) by breaking tasks into sub-steps, each of which may be more easily recovered than the original task. The user should be informed of the previous state to which the system will revert. Hence, Trend and Notification may help users execute Recover.

A.17 Redundant Information

Context
The user is required to perceive and interpret information and act on that information in maintaining the safe function of the system. The information is complex, or could be misperceived or misinterpreted.

Problem
How can we enhance the likelihood that the user correctly perceives and interprets safety-critical information?

Forces

- Some information is more salient than other information.
- Safety-critical systems may be complex, with large amounts of information that needs to be available to the user.
- The amount of display space available is limited.
- Providing too much information will overload the user; the information that is displayed needs to be chosen carefully.

Solution
Provide more than one way of viewing information so that the likelihood that it is misunderstood or ignored is lessened. Redundancy in user interface designs can be viewed as an extension of the usual safety-critical viewpoints advocating redundant hardware and "n-version programming" (but note that such approaches are often unsuccessful in software design). The field of communication studies has looked at situations such as combination of audio and visual cues in television (e.g., both auditory and visual warnings). Redundant views should be observably consistent.

Examples
The use of the International Phonetic Alphabet ("Alfa, Bravo, Charlie..." instead of "A, B, C...") [15] is a form of redundant input that has been in use for many years to improve human-human communication. Similarly, the purpose of levers can be more readily comprehended if there are adequate cues beyond just a label, e.g., shape, position, colour (see [15]). The Defibrillator system uses both written and spoken alerts to inform the user of the system's state.

Resulting Context
Information that is vital to safety will be displayed to the operator in several redundant forms. Redundant Information will usually take the form of an Abstract Mapping.

A.18 Separation

Context
The system provides several actions for which the corresponding physical executions are very similar. Alternatively, several information displays are provided for which the layout and presentation are very similar. When one is appropriate, the other is not and possibly vice versa. In addition, it is not possible to predict hazards and prevent the potentially hazardous action.

Problem
A system is constructed from components that limit the scope for distinct executions and presentations (e.g., a graphical toolkit), so that the potential for confusion between components in different contexts arises. **How can we reduce the likelihood that users will inadvertently perform the wrong action or misinterpret information?**

Forces

- We would like to reuse components because modern systems usually incorporate graphical interfaces and it is impracticable to not use them where safety will not be compromised.
- Even if a toolkit is custom-built, the widgets and interaction mechanisms must then be reused in the design.
- Systems are built within a budget.
- When widgets are reused, unless the customisation is extensive, components will often appear similar to users.
- Reusing commercial components means we cannot as easily customise them.
- The potential for user error increases as the similarity and proximity of controls increases.

Solution
Separate two controls (physically or logically) if they are operated in a similar way.

Examples
Most style guides recommend separation for distinct operations that are accessed by similar controls (e.g., most Windows programs separate the "OK" and "Cancel" buttons in dialogues). The Defibrillator system separates the on/off button from the Analyse and Shock buttons to reduce the chance of the user accidentally turning the unit off when a shock was intended. The Angiography machine separates the 5-minute and hour timer reset buttons to reduce the likelihood of the operator pressing the hour reset button rather than the 5-minute reset button.

Resulting Context
System components that are operated by similar physical actions are physically or logically separated. Affordance and Distinct Interactions may also be used to reduce operator execution errors.

A.19 Shutdown

Context
Shutting down the system leads to a fail-safe state and the service provided by the system can be halted for at least a temporary period, and:

- failure can lead to a serious accident within a very short time;
- shutdown is simple and low cost; and
- reliable automatic response is possible.

Problem
How can safety be assured if a serious hazard has occurred?

Forces

- In the event of component failure, a safety-critical system may continue to function, with graceful degradation of service. However such a malfunctioning system is inherently less safe than a fully-functional system.
- Systems such as factory plant can often be shut down in a safe state.
- Systems where failure of service is a hazard usually cannot be shutdown, e.g., medical life-support devices, aircraft, air traffic control services, missiles, etc.

Solution
When shutdown is simple and inexpensive, and leads to a safe, low risk state, the straightforward solution is to shut down automatically.

When a system cannot be shut down automatically because of cost or complexity, then the system must instead be stabilised, either manually or automatically [4]. If a failure can lead to a serious accident within a very short time without shutdown, then reliable automatic response is necessary, and if this is not possible, then the system should not be built if the risk is unacceptable.

Many medical systems involve the preservation of life, saving lives, or perform processes which if left incomplete, may result in loss of life (e.g., dialysis). Such machines cannot be safely shut down and the correct response when a malfunction occurs is to attempt to continue operation albeit in a degraded mode.

Examples
McDermid and Kelly [19] give an example of an industrial press that automatically moves to a safe failure state (i.e., press closed) when the sensors for press movement are inconsistent. Medical treatment devices such as the Magnetic Resonance Imaging system and the Angiography system can be safely shutdown without harm to the patient.

Design Issues
Shutdown should be ennunciated by an Alert to the user that the system is no longer operating.

Resulting Context
The system shuts down on detection of failure. If a system cannot be simply shutdown, then Automation might be used to stabilise the system and Notification is indicated to bring the failure to the user's attention. If the Shutdown of the system will not completely alleviate the problem, then an Alert should also bring the user's attention to the situation.

A.20 Stepladder

Context
The system is defined by a set of tasks that are decomposed into logically simpler tasks and the effect/consequence of incorrectly performing a task cannot be readily diminished.

Problem
How can we guide the user through complex tasks?

Forces

- It is desirable for the user to remain familiar with low-level tasks, so they are capable of dealing with novel situations.
- When performing a complex task, it is easy for users to forget what they have and have not done. This is especially true when there are other distractions.
- Users may eventually see the task sequence as a single, aggregate, task.

Solution
Identify complex tasks and explicitly split them into sequences of simpler tasks. In some cases, the task sequence may form a new task itself. For example, a Wizard in Microsoft Windows is considered a separate task which enables several smaller tasks to be performed in a sequence. In other cases, there is no explicit representation, it is simply a design consideration, which has led to the creation of several individual tasks. Even in this case, though, the user's tasks may be controlled by the system's mode, and Behavioural Constraints and Affordance can be applied to help the user identify what task comes next.

Examples
The concept of explicit procedures is well-established in safety-critical systems design. Aircraft crew are provided with reference cards explaining step-by-step what to do in emergencies. Sometimes, controls on machines are arranged in a way which suggests a certain execution order. As an example from the non-safety-critical domain, Melbourne's public transport ticket machines require the user to specify three variables (zone, concession, expiry time), place money in the machine, and then collect the ticket. Even though the three variables can be entered in any order, the design of the machine suggests a particular order, arbitrary though it may be. The overall left-to-right ordering of controls provides an Affordance as to the appropriate sequence and suggests to users what to do next.

In the Hypodermic Syringe example interface, in the usual case, several simpler actions are required to equate to the corresponding action when using a keypad. The positioning of the +/- buttons affords the appropriate sequence. The Defibrillator breaks down the complex task of giving someone an electric shock to restart their heart into discrete steps. Each step leads to another action for the user to perform. The system also prompts the user at each step with the next action.

Resulting Context
After applying this pattern, the user should have an improved idea of what tasks need to be performed at any given time. The pattern works best in tandem with Transaction and Notification. Each few rungs of the stepladder form a Transaction. This way, many of the individual tasks will be easily recovered (Recover) from, because they will not be conveyed immediately to the broader system. By splitting the original task into subtasks, the consequence of each step may be less than for the original task and Recover may become easier to apply. The Stepladder can be used to structure Affordances.

A.21 Task Conjunction

Context
A task has been designed which has a relatively high risk of producing a hazardous state and errors cannot be prevented, e.g., because the task involves data-entry. The task is not time critical.

Problem
How can we check whether a user's action matches their intention?

Forces

- Redundancy is widely used in the safety industry to avoid hazards arising due to a component of the system failing. The system is said to have no single point of failure.
- Entry fields, or screens in a user interface can be regarded as components of the system.
- Operators are likely to make errors on simple or repetitive tasks such as data entry.

Solution
Reduce errors by requiring that the user perform tasks multiple times. The user's performances on each iteration of the task are compared and the outcome used to determine whether the task has been correctly performed. Redundant tasks are an error detection technique. Redundancy reduces the efficiency with which users can perform tasks and therefore the "raw" usability of the system, but often enhances system safety by enabling error detection. Another variant is requiring the same task to be performed by two different users, as in a detonator, which can only be activated by two people.

The Task Conjunction pattern is similar to Transaction, in that it requires several actions before any commitment is made. However, the intention differs. In Task Conjunction, there is only one fundamental change, but it is subject to verification actions. In Transaction, each action provides *new information* about desired changes. The conjunction must not be able to be circumvented, e.g., on the Therac-25 machine, users could press "Enter" to confirm a value rather than re-type the value. Pressing "Enter" soon became automatic for the users [16].

Examples
Redundancy in software has a long history. Communication protocols, for example, use checkbits and other data to enable error detection and/or correction. Redundant procedures, in which a supervisor checks work performed by others, are commonly used to reduce error in safety-critical systems. For example in the Dye Delivery system, a second nurse checks the settings entered by the operator of the system.

Resulting Context
The original task is redefined as a conjunction of redundant sub-tasks, to which Transaction may be applied.

A.22 Training for uncontrolled environments

Context
The environment in which the system is used is not controlled. This environment may be the physical situations in which a system is operated or more difficult to define situations involving the subtle difference between cultures or organisational structures. Thus limited or incomplete controls can be placed within the system to ensure safety.

Problem
How can we ensure safety when we have little or no control over the environment?

Forces

- The user is familiar with the existing environment.
- Operators need to train regularly to avoid de-skilling.
- Environmental factors exterior to a system are not always obvious.
- Environmental factors are out of the designer's control.

Solution
One way to deal with this situation is to **provide users with sufficient training to allow them to operate the system in a manner that has the lowest risk.** It is also very important to ensure that de-skilling does not occur, that is, a user does not lose proficiency with a system after a period of non-use. It is important to realise that this pattern is not the optimal solution to this problem and should only be applied if all other patterns have proved to expensive or difficult to implement.

Examples
Many systems in the safety-critical realm rely on training to help decrease the risk of hazards. In the defibrillator example the user must be trained in the use of the system to increase their understanding of the dangers involved. The notion of conductivity is not one that is always obvious. For example, the patient may be lying in a puddle of water, or they may be lying on a wet pool deck. As the life-pack has the potential to send a massive electric shock through the human body (it is, in fact, designed to do this), it is important that the user not be in physical or conductive contact with the patient in any way. The system has no fail-safes or methods to ensure that no person, except the patient, comes into contact with the electric current during operation except through the use of the motion detector. The Lifepack provides training modes in one of two ways. Firstly, if the defibrillator can be taken offline for the purpose of training then a small data card can be inserted into the system to inoculate the electrodes (i.e., no shock can be given). The interaction will continue from there in exactly the same fashion, without any risk of accidental electrocution. Secondly, through the provision of a stand-alone simulator that looks somewhat different to the real defibrillator (to prevent confusion in an emergency).

Resulting Context
Excessive use of this pattern can lead to an unsafe system by overly relying on the human operator. It is important that after the user has been trained that the risks are assessed to see if they are at an acceptable level. To prevent de-skilling it is important that frequent reviews and retesting occur. To aid in testing it is imperative that users be trained on the system itself or a high fidelity simulator if the system itself cannot be taken offline to perform the training. The designer must remember that this pattern is only a last resort; it is by no means the safest way to implement a system. However, when costs are too high or the effort involved would be unacceptable, then this pattern may be applied sparingly.

A.23 Training for uncontrolled interfaces

Context
A new functional core is being introduced into an existing system that has an existing interface. Thus limited or incomplete controls can be placed within the system to ensure safety.

Problem
How can we ensure safety when we have little or no control over the interface of a system?

Forces

- The user is familiar with the existing system's interface.
- Operators need to train regularly to avoid de-skilling.
- Reuse of an interface is inexpensive but implies acceptance of the existing controls and checks of the system.

Solution
One way to deal with this situation is to **provide users with sufficient training to allow them to operate the system in a manner that has the lowest risk.** It is also very important to ensure that de-skilling does not occur, that is, a user does not lose proficiency with a system after a period of non-use. It is important to realise that this pattern is not the optimal solution to this problem and should only be applied if all other patterns have proved to expensive or difficult to implement.

Examples
Many systems in the safety-critical realm rely on training to help decrease the risk of hazards. Where the interface remains the same, but the functional core of the system changes, the problems are difficult to deal with. For example, the Therac 25 radiation therapy machines attempted to combine the functionality of the Therac 6 and 20 machines into one machine. The earlier machines were run with hardware interlocks in place to prevent overdosing; the computer components were merely to aid in ease of operation. In fact, both systems were capable of being run entirely without the computer. In the Therac 25 the system removed most of these interlocks and attempted to replace them with software interlocks. During the 2 years Therac 25 was in operation the machine caused 6 major incidents, some of which resulted in death. All of the accidents were due to faults in the software that caused the amount of radiation delivered to be grossly understated. This problem arose because the users of the system were used to using the old machine and when the new one was

brought into service with an almost identical interface the users just assumed that it was operated in the same manner. When the interlocks were removed from the system, some errors in the program that were unnoticed before became a major issue due to the interlocks no longer preventing these errors. After these accidents the developers recalled the machines and retrained the users to ensure that they were aware of the problems in the systems operation. So after the accidents the underlying functional core was again modified but the interface remained the same.

Resulting Context
Excessive use of this pattern can lead to an unsafe system by overly relying on the human operator. To prevent de-skilling it is important that frequent reviews and retesting occurs. To aid in testing it is imperative that users be trained on the system itself or a high-fidelity simulator if the system itself cannot be taken offline to perform the training. When considering application of this pattern the designer must take into account the options that they have in making the functional core of the system more "bulletproof" by including machine control. It is important that after the user has been trained that the risks are assessed to see if they are at an acceptable level. It is important to remember that this pattern is only a last resort; it is by no means the safest way to implement a system. However, when costs are too high or the effort involved would be unacceptable, then this pattern may be applied sparingly.

A.24 Transaction

Context
Actions are not time-critical and hence can be "stored-up" before being applied and:

- the effect of the task as a whole is difficult to undo;
- sub-steps can be undone; and
- risk is relatively low compared to cost and difficulty of prevention.

Problem
For many real-time systems, it is difficult to provide a Recover action which has any practical value because the system usually changes irreversibly by the time the user tries to recover from the unwanted action. **How can we improve recoverability?**

Forces

- It is relatively easy to recover tasks that do not impact on real-world objects.
- Often reversal is useful to iteratively define a task's parameters.
- Transactions are used in data-processing to enable the effect of a sequence of actions to be "rolled-back" to the state at the commencement of the transaction.
- Transactions bundle a sequence of task steps into a single task. Hence, they are ideal for structuring interaction in terms of overall goals and sub-tasks.

Solution
Bundle several related task steps into a transaction, such that the effect of each step is not realised until all the task steps are completed and the user commits the transaction. By grouping task steps in this way, it becomes very easy to Recover the effect of a sub-step before the transaction as a whole has been committed. In addition, because errors are deferred until the transaction is committed, users have more time to consider their actions and to recover from them if appropriate.
Each transaction should involve task executions and information that is physically grouped on the users console or display. For example, a data entry transaction might be implemented as a pop-up window with commit and abort buttons to either accept or reject the information entered by the user.

Examples
A standard dialogue box supports this pattern. The user can easily enter data via text fields and buttons, but none of these choices matter until they hit a confirmation button (e.g., one labeled "OK"). The Angiography and Dye Delivery systems use forms to receive user input. The input process can be aborted at any time before the form has been completed.

Resulting Context
Task steps are grouped into transactions with commit and abort options for each group. The commit step in a transaction can quickly become automatic for the skilled user which can be countered by applying Task Conjunction. If it is appropriate for the transaction's sub-tasks to be constructed iteratively, then the transaction can be viewed as a form of Stepladder. A sequence of transactions themselves can also form a Stepladder.

A.25 Trend

Context
Users need to formulate and follow task plans that involve attention to the change in state of the system, e.g., where an action must occur if the state is in a certain configuration, or when a state change occurs.

Problem
How can the user be notified that the state has changed (i.e., the trend of the system is towards a hazardous state)?

Forces

- Many user errors stem from memory limitations. Users may not notice that the state of the system has changed and that they should take action.
- Memory-based errors may occur even when the user has previously formulated a plan to perform a particular action when the state of the system reaches a particular configuration. For example, in air traffic control, the user may need to change the altitude of an aircraft before it reaches a particular waypoint, but may not be able to do so immediately because of other more pressing concerns. A hazard situation arises when the user fails to return to the original aircraft and change its altitude, after resolving the immediate concern.

Solution
Allow the user to compare and contrast the current state with previous states. This will help users assess the current situation and formulate plans for the future.

Examples
The Druide air traffic control system displays aircraft as a trail of locations with the most prominent location displayed being the immediate location. Oil pressure gauges in aircraft may display a shaded region that indicates the oil pressure in the previous 5-minute interval. An automatic syringe could display the amount remaining in the syringe and the amount remaining at set prior intervals. A medical system for controlling an oxygen pump and mask might use Trend to indicate the oxygen content of the delivered gas now, and at a prior point of time.

Design Issues
One common technique is to overlay previous states on the same display, showing the previous state in a less visually striking manner (e.g., muted colours). This is particularly effective for motion, as a trail of past motion can be formed. If this technique causes too much clutter, a replica of the past state can be placed alongside the current state. This, however, occupies valuable screen real estate, and may hamper direct comparison.

Resulting Context
State changes are explicitly displayed to the user. Display of the state change is a Reality Mapping. The change in state may also be brought to the user's attention via a Notification, if it has a readily identifiable safety implication. When a system Trend is indicating decay into a hazard state an Alert should be raised.

B Example Interfaces

In this appendix we give summaries of the exemplary user interfaces that have been used as examples for the patterns described in Section 3.3 and Appendix A.

B.1 Defibrillator

The Defibrillator [25] is designed for use by first responders to cardiac emergencies. The system is extremely portable and was designed with the infrequent user in mind. The Defibrillator unit consists of two electrodes and the battery pack/control unit (as shown in Figure 4). It uses a shock advisory system to inform users how to proceed. ECG data and on-scene audio can be stored digitally within the device to allow full incident reporting.

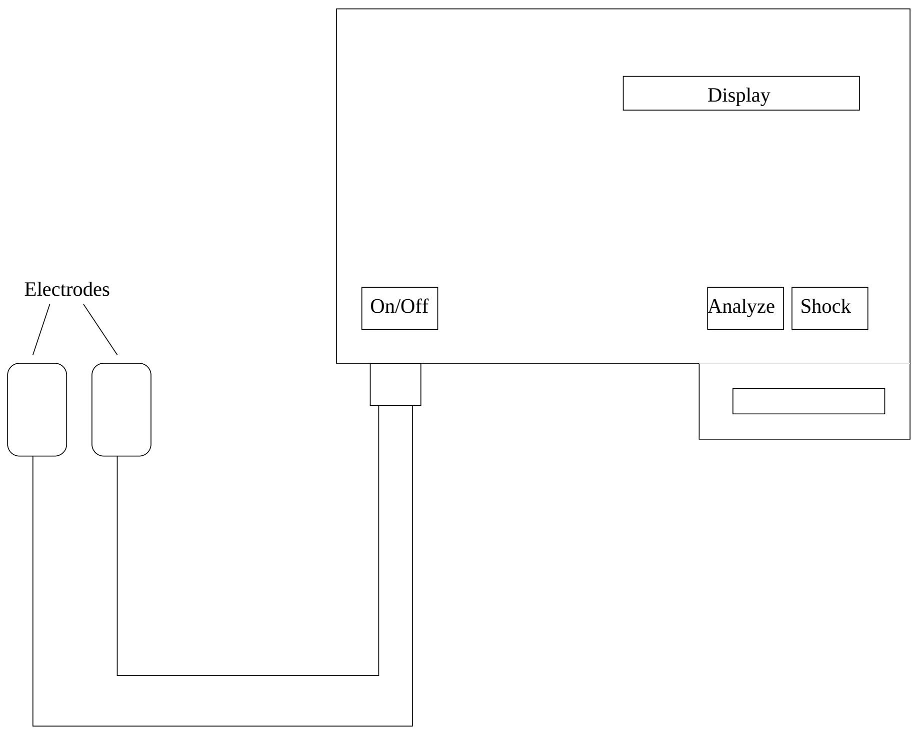


Figure 4: Defibrillator user interface

The defibrillator controls are simple: the three main buttons are “on/off”, “analyse”, and “shock”. There is also a switch provided to set the clock but this switch has been left off the diagram for simplicity. The on/off button is self-explanatory. The analyse button, when pressed, begins ECG analysis. The shock button is only active when the shock advisory system permits defibrillation. The display is a 20-character dot matrix liquid crystal display. If the low battery indicator is lit the system has at least eleven discharges remaining with a non-rechargeable battery pack, and at least six discharges remaining with a rechargeable battery pack.
Typical operation of the system proceeds as follows:

1. Turn on the defibrillator.
2. The system will inform the user to place the electrodes on the patient.
3. Once the electrodes have been placed, the system will advise the user to press the analyse button.
4. The system performs a series of tests based on the ECG reading of the patient.
5. Depending on the outcome of these tests, the system will further prompt the user.
   (a) If the system believes that a shock is required it will prompt the user with the following series of outputs: “shock advised”; “send clear”; “push to shock”.
   (b) If the system believes a shock is not required it will prompt the user with the following outputs: “no shock advised”; “check for pulse”; “if no pulse, start CPR”.
   (c) If the system detects movement of the patient during the analysing procedure it will provide the user with the following outputs: “motion detected”; “stop motion”.
6. If at any time during operation the system detects a low battery it will prompt the user with: “replace battery”.

The system prevents a shock being given if it detects a pulse or if movement is detected. All of the display prompts are also spoken by the machine to reduce the chance of user confusion. The system is unable to ensure that no one but the patient receives an electric shock. This can only be achieved by ensuring that the operator has been correctly trained.

B.2 Hypodermic Syringe

Dix [7, page 6] describes an automatic (computerised) syringe. The primary user task is to enter a dose for the syringe before applying the device to a patient. The original user interface for the system had a calculator style interface that enables doses to be rapidly entered (see Figure 5 (a)). However, because the syringe could be injecting pharmaceuticals that are lethal outside a safe range, the original design does not sufficiently consider risk. When risk is taken into account a better design is given in Figure 5 (b). In the modified design, the user cannot enter doses as quickly and more effort is required to do so (so usability is reduced). However, the system is safer because a single extra key press is less likely to produce an order of magnitude dose error. Additionally, the modified system provides error tolerance by allowing the dose to be changed (which also enhances usability).

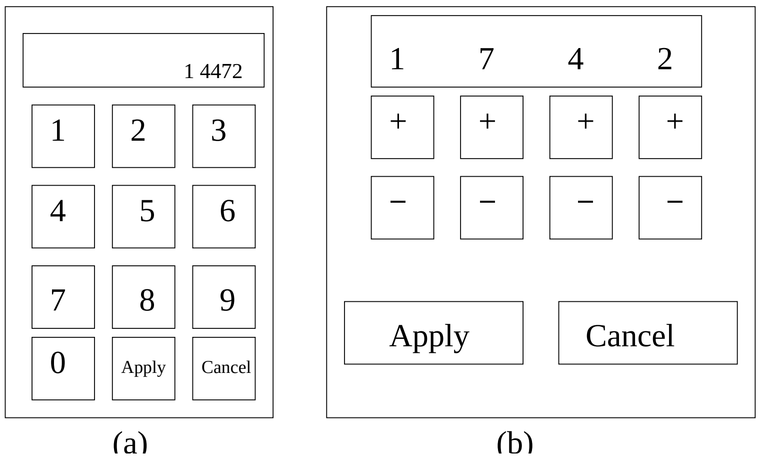


Figure 5: Syringe design: (a) original; (b) revised (adapted from [7, page 6])

B.3 Neurosurgical Planning

This system is a three-dimensional user interface for pre-operative neurosurgical planning based on the physical manipulation of familiar real-world objects in free space [9]. Neurosurgeons can apply their existing skills to specify spatial relationships between the skull and utility "props" in a natural and direct manner. The props are real-world objects that approximate the virtual object on screen. The interface employs a head viewing prop, a cutting-plane selection prop, and a trajectory selection prop. Each prop is a simple real-world tool; the computer tracks the position and the orientation of the skull for each prop. From the surgeon's perspective, the interface is analogous to holding a miniature skull which can be "sliced" and "pointed to" using the cutting-plane and trajectory props. The system can be controlled by either speech or gesture. The surgeon's speech can be recognised using voice-input technology, and his or her gestures are monitored by having them manipulate props, the position and orientation of which are tracked by the computer. The props are tracked using the Polhemus FASTRAK six-degree-of-freedom digitiser. To manipulate the computer model of the patient's head, the physician is provided with a solid rubber ball that can be held comfortably in one hand. The ball can be turned to rotate a polygonal model of the patient's brain, and can simultaneously be moved towards or away from the user's body to control the zoom factor. A cutting-plane prop is also provided, which can be used to specify arbitrarily oriented slices through the patient's anatomy, allowing the surgeon to examine cross-sections of the volume or to dissect obstructing portions of the anatomy. The cutting-plane is used in conjunction with the head prop rather than as a separate tool. The user positions the head prop with their non-dominant hand while holding the cutting-plane up to it with their dominant hand. A trajectory selection prop allows the surgeon to specify three-dimensional vectors and points. The trajectory prop is a stylus-shaped tool, which is equipped with a tip switch. Moving the trajectory prop relative to the head prop specifies the position and orientation of a cylindrical virtual probe relative to the polygonal brain model.

B.4 Angiography

The Angiography system was documented from personal communication with staff of the Royal Melbourne Hospital. Input to this system is by a touch screen interface. The machine takes multiple images and uses image subtraction to produce the information. Images can be produced at a rate of up to 6/second or as few as 1 in 5 seconds. The operator presses the button of interest, which highlights the appropriate input field, and uses a "one-finger-to-rotate" dial to change the value for that field. This machine allows the user to select a particular kind of scan template (e.g., cranial arteries). There is a template with initial values for each body-part. The operator alters the template settings to set the value required. There is a timer, which has to be manually reset every 5 minutes and which terminates the scan if not reset, and this can continue for up to an hour. The operator presses a button to reset the timer. When the timer reaches an hour scanning ceases and there is a continual buzzing until pressing a second button resets the machine. The operator is notified that they have been scanning for an hour. If the operator presses the hour reset button rather than the 5-minute reset button, scanning may continue for more than an hour. The two buttons are separated by a fair amount of screen real estate. It would not be likely that an operator would press the wrong button.

B.5 Dye Delivery

The Dye Delivery system was documented from personal communication with staff of the Royal Melbourne Hospital. This system is of a similar design to the infusion syringe. The main difference is that input is taken via a calculator-style numeric keypad. As in the Angiography system, templates for the various parts of the body give an upper and a lower limit (preset for each part of the body) to the amount that can be delivered. In this case, the dye is delivered into the bloodstream. The delivery of the dye is thus directly into the body part requiring observation via a catheter. To reduce the likelihood of incorrect settings, a second nurse checks the settings entered by the operator of the system.

B.6 Magnetic Resonance Imaging

The Magnetic Resonance Imaging system was documented from personal communication with staff of the Royal Melbourne Hospital. The Magnetic Resonance Imaging (MRI) machine scans in preset "slices". When the operator is presented with a scan, reconstructions (image processing) have been performed. The most serious safety issue affecting this machine's operation concerns the radio frequency (RF) used. The RF input into a patient warms them. There is a monitor for patient temperature at the scan site. When the detected heat has reached a critical level the machine ceases to scan. Overheating is only really a problem if too many of the same images of the same type are scanned.

C Interaction description for Radiation Therepy Machine

This appendix covers the detailed design and steps through the possible interactions of the design.

The initial screen of the user-interface is shown in Figure 6. 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.

For ease of description the various interaction areas of the interface have been labelled. The three areas are:

- the *preview panel*;
- the *data entry panel*; and
- 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[4] to check the input data, large discrepancies will be obvious and the user will be able to investigate these discrepancies. All data for the treatment is to be entered into the *data entry panel.* Any interaction that occurs within this panel is going to effect 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.*

[4]In this design the user or operator is the radiologist operating the machine.

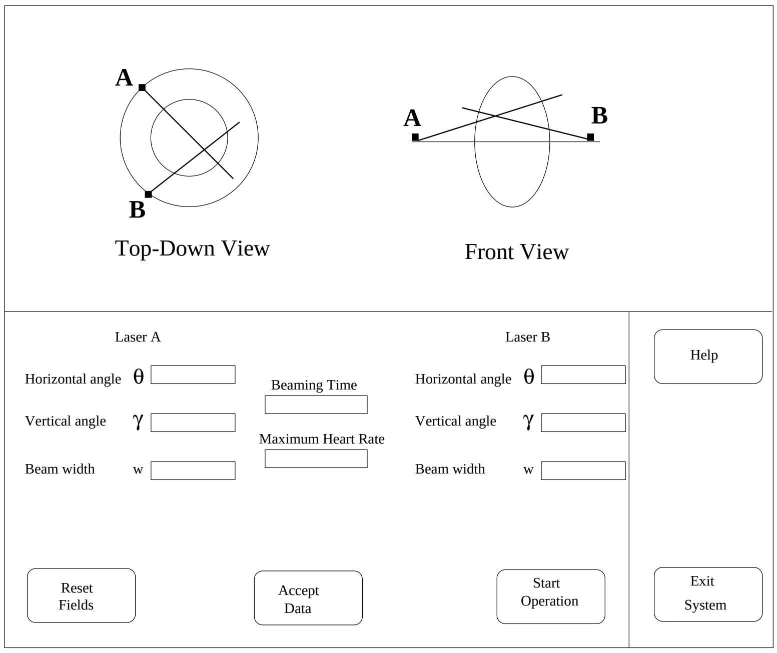


Figure 6: The initial user-interface

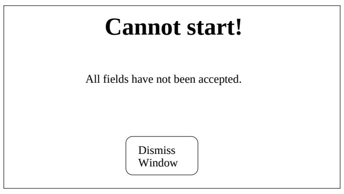


Figure 7: Unaccepted fields error.

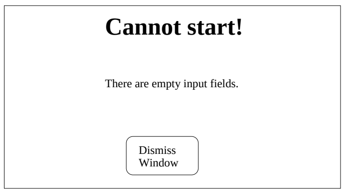


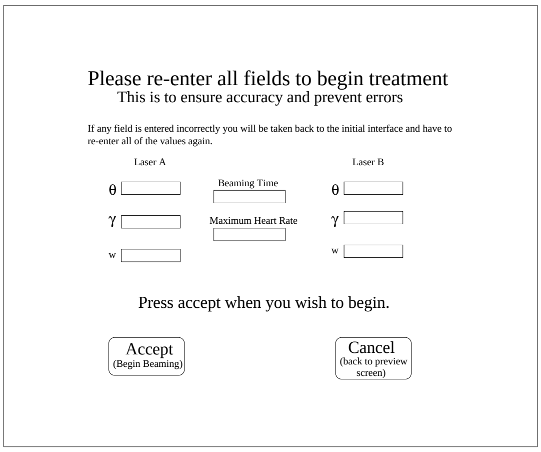
Figure 8: Empty fields error.

In the *system panel* if the *help button* is pressed, a *help window* comes up that contains a table of contents and an index. This document is navigated via a hypertext interface. Pressing the *exit system button* will close down the beaming application. Most of the interaction will occur within the *data entry panel.*

Before treatment may commence the user must enter the data into every field within the *data entry panel.* Once the data is entered into the fields, the user can press the *accept data button* or the *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 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; this inability is obvious to the user as the button is "grayed out". The user may now press the *Start button*, the *reset button*, or edit some or all of the fields. If the user attempts to press *Start* without first accepting any changes made since the last accept event, they will be shown the *error window* in Figure 7. If the user attempts to press the *Start button* when there are empty input fields the *error window* in Figure 8 will be displayed. In both cases the *error windows* will be accompanied with a critical stop sound event.

Once the *Start button* has been successfully pressed (no erroneous conditions) the user is shown the *confirm screen* given in Figure 9. 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 user presses the *Accept button*, and all fields are accurate, the machine will begin beaming and present the *beaming interface* to the user (shown in Figure 10).

The *beaming interface* contains two information panels and the *emergency panel.* The top panel contains all of the information that was previously supplied by the interface, as well as the monitoring information of which a user must be aware. The *emergency panel* indicated in Figure 10 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.


Please re-enter all fields to begin treatment
This is to ensure accuracy and prevent errors

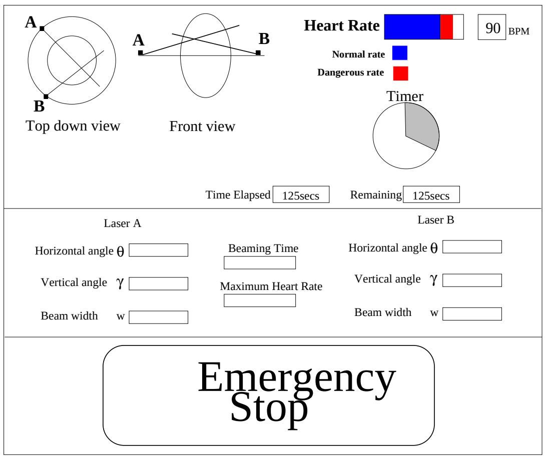
If any field is entered incorrectly you will be taken back to the initial interface and have to re-enter all of the values again.

Laser A | Laser B
θ | Beaming Time | θ
γ | Maximum Heart Rate | γ
w | | w

Press accept when you wish to begin.

Accept (Begin Beaming) | Cancel (back to preview screen)

Figure 9: The Confirm screen.



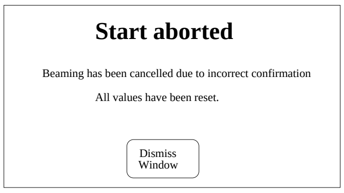
Heart Rate 90 BPM
Normal rate 🟦
Dangerous rate 🟥

Time Elapsed 125secs | Remaining 125secs

Laser A: Horizontal angle θ; Vertical angle γ; Beam width w
Beaming Time
Maximum Heart Rate
Laser B: Horizontal angle θ; Vertical angle γ; Beam width w

Emergency Stop

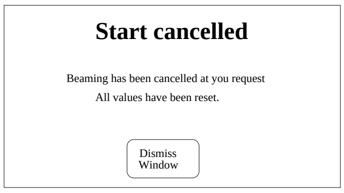
Figure 10: The Beaming Interface.


Start aborted

Beaming has been cancelled due to incorrect confirmation

All values have been reset.

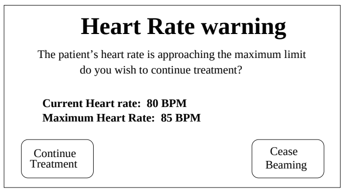
Dismiss Window


Start cancelled

Beaming has been cancelled at you request

All values have been reset.

Dismiss Window


Heart Rate warning

The patient's heart rate is approaching the maximum limit do you wish to continue treatment?

Current Heart rate: 80 BPM
Maximum Heart Rate: 85 BPM

Continue Treatment | Cease Beaming

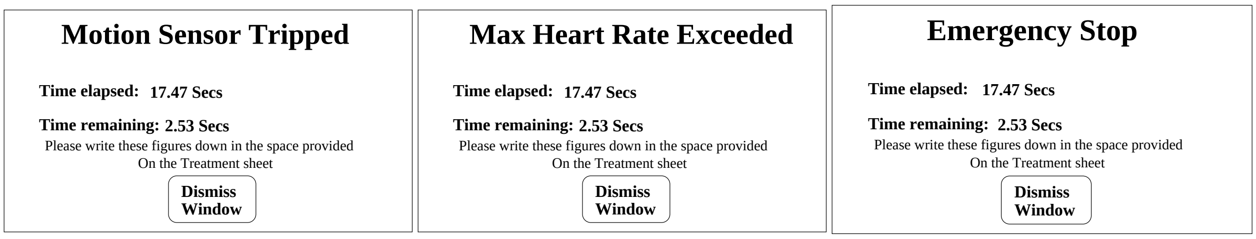
Figure 11: Aborted start. Figure 12: Cancelled start. Figure 13: Heart rate warning.

During operation the user will be unable to change the entry fields contained in the data panel, and thus the panel will be "grayed out" to show the user that the fields are unmodifiable[5]. If the information entered in the confirm screen differs from that entered on the initial interface, the system will display the initial user interface again with all fields cleared, and a dialog box informing the user that the start has been aborted due to incorrect confirmation (Figure 11). If the user wishes to cancel the confirmation and return to the main user interface without starting machine operation, they may press the *cancel button*. When this button is pressed the system will return to the main user interface, with all fields cleared, and a dialog box informing the user that the confirmation has been cancelled (Figure 12).

During operation of the machine 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 a dialog box that informs them that the patient's maximum heart rate is being approached (Figure 13). This window will dismiss itself after twenty seconds. Any event that occurs while this window is present will take precedence, which 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 user that the system requires their attention. If the user does not see the screen during the 20 seconds[6] then the alarm will inform them that the patient's heart rate is approaching dangerous levels. The system will continue to alert the user at a random interval between two and three minutes that the heart rate is near the maximum by reissuing the error window, until the patient's heart rate drops, exceeds the defined maximum or the system completes beaming.

If an emergency stop occurs during the system's operation, a window informing the user of the type of emergency stop[7] will be raised. This window (regardless of type of stop) will contain the time elapsed and time

[5]No graying out is shown in the diagrams to maintain image clarity.
[6]This is unlikely, but possible.
[7]The types of emergency stop are: *emergency stop button* pressed, patient motion detected, or the heart rate reaches the preset limit.


Motion Sensor Tripped

Time elapsed: 17.47 Secs

Time remaining: 2.53 Secs

Please write these figures down in the space provided
On the Treatment sheet

Max Heart Rate Exceeded

Time elapsed: 17.47 Secs

Time remaining: 2.53 Secs

Please write these figures down in the space provided
On the Treatment sheet

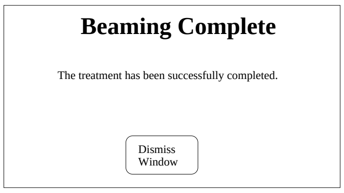
Emergency Stop

Time elapsed: 17.47 Secs

Time remaining: 2.53 Secs

Please write these figures down in the space provided
On the Treatment sheet

Figure 14: The three instances of the emergency stop error window class.


Beaming Complete

The treatment has been successfully completed.

Figure 15: Beaming complete window.

Sound event	System event (labelled by figure associated with event)
Critical stop alert. (cymbal crash)	7, 8, 11 and, 12
Heart rate alarm	13
Emergency stop alarm	14
Single gong sound	15

Table 2: Sound event table.

remaining at the instance of stopping. The user must record this information then dismiss the window, to ensure that there is a record of how much radiation a patient has received if there is a need to resume the treatment. The initial user interface screen will be shown with all fields empty. The three types of error window are shown in Figure 14. If the system reaches the end of its beaming period the data entry and preview panels will be cleared and a message window will be displayed informing the user that the treatment has been successfully completed. This final dialog box is shown in Figure 15. Table 2 shows what system events are accompanied by a sound event and provides details of the nature of that sound.