Formal Specification of User Interface Design Guidelines

An Overview

Judy Bowen

July 2004
Background

- What?
- Why?
- How?
What? - Formal Specification

- We specify the system behaviour
- We validate our specification
- We show a refinement to our implementation
- We test the code and prove the correctness of our system

And then?
A user comes along and breaks it
What? - User Interfaces

- Human Computer Interaction
- Make the system work for the user
- If the user can successfully interact with the interface
- And the interface successfully interacts with the system
- And the system is correct
The world is a better place
What? - Design Guidelines

- General properties of interfaces that make them more usable
- May be domain specific
- May be widget driven
- May be aesthetic - house styles, platform guidelines
Why?

- Despite the best efforts of HCI practitioners, we’re still getting it wrong.
- It’s not just a design issue.
- It’s not just a usability testing issue.
- It’s about getting more things right earlier in the software lifecycle.
Why?

- Safety critical systems e.g. Nuclear power plants, industrial plants
- Sanity critical systems e.g. The tools used worldwide everyday
How?

- Include the interface in our system specification
- Include design guidelines in our interface specification
- Prove that the specification implies a usable design
Some Background

- Pixel level definitions
- PIE
- Interactors
- Temporal Issues
- Algebraic Methods
Some Problems

- Choosing the right guidelines for the domain
- Mapping the levels of abstraction
- Maintaining the readability of the specification
- Getting the designers to understand / use it
So Far

- Different ways of formalising interfaces
- Formalising widgets
- Different levels of abstraction
- Refinement Issues
- Support tools
A Small Example

- Modelling a system which has user controlled display options
- User can select from one of three choices
- Choices determine the size of the current window display
Operation and Widget Schema

```
Option := Selected | NotSelected
WNDISPLAY := Max | Half | Min

DisplayViewChange

Delta Window
displayType? : WNDISPLAY
display? = displayType?
level? = level
default? = default
```

```
DisplayViewWidget

fullscreen : Option
halfScreen : Option
scrollPanel : Option
displayType : WNDISPLAY

fullscreen = Selected ∨ halfScreen = NotSelected ∨ scrollPanel = NotSelected
halfScreen = Selected ∨ fullscreen = NotSelected ∨ scrollPanel = NotSelected
scrollPanel = Selected ∨ fullscreen = NotSelected ☼ halfScreen = NotSelected
fullscreen = Selected ⇒ displayType = Max
halfScreen = Selected ⇒ displayType = Half
scrollPanel = Selected ⇒ displayType = Min

ActiveDisplayViewWidget ⊑ DisplayViewWidget ⊚ DisplayViewChange
```
First Prototype
A Problem

- Refinement shows we cannot prove the mutual exclusivity of the options about this interface
- User testing shows the system breaks when a user selects more than one option
- Designer fixes it
Second Prototype

[Image of a dialog box with options: Full, Half, Panel]
But ..... 

- Isn’t this the original prototype?
- Designer has “improved it”
- User can now only select one check box
- Designer has broken guidelines regarding selection controls
Guidelines for using Selection Controls

- Use radio buttons to indicate one or more options that must be either on or off, but which are mutually exclusive.

- Use checkboxes to indicate one or more options that must be either on or off, but which are not mutually exclusive.
Extending the Specification

- Design must satisfy our specification
- Design must also satisfy guidelines
- Find a way to specify selection widget guidelines
- Ensure the described property holds in our system
Extending the Specification

\[ \exists \text{ system} : \text{SystemState}, \text{ebOne} : \text{CheckBox}, \text{ebTwo} : \text{CheckBox} \ni \\
\quad \text{ebOne} = \text{Selected} \land \text{ebTwo} = \text{Selected} \]

\[ \exists \text{ wi : DisplayViewWidget } \ni \text{wi.fullScreen} = \text{Selected} \land \\
\quad \text{wi.halfScreen} = \text{Selected} \land \text{wi.scrollPanel} = \text{Selected} \]

\[ \forall \text{ rbOne : RadioButton}, \text{rbTwo : RadioButton} \ni \\
\quad \text{rbOne} = \text{Selected} \iff \text{rbTwo} = \text{NotSelected} \]

\[ \forall \text{ wi : DisplayViewWidget } \ni (\text{wi.fullScreen} = \text{Selected} \iff \text{wi.halfScreen} = \text{NotSelected}) \land \\
\quad (\text{wi.fullScreen} = \text{Selected} \iff \text{wi.scrollPanel} = \text{NotSelected}) \land \\
\quad (\text{wi.halfScreen} = \text{Selected} \iff \text{wi.scrollPanel} = \text{NotSelected}) \]
The Revised Prototype

Select Screen Display

- Full
- Half
- Panel

Making it Better

- Adding visual guidelines

\[ \forall \text{options : DisplayOptions} \ni \text{hasVisualClue} = \text{true} \]
The Final Prototype
Finally

- Questions?
- Comments?