Gary Marsden

gaz@cs.uct.ac.za

2006
Putting users first

- Software development methods tend to focus on technology - You end up with a bunch of functions that the software can deliver
So much for UCD

- This forces users to panic, give up, attend training courses or open a shop called “crystals ‘n’ candles”
- With personas we know who will be using the software
- We need scenarios to help us think about what they will do with it.
Remember the cow!

- This is often the thing computer scientists feel is ‘wrong’ with HCI
  - Hang in there for a moment
- The idea of scenarios is to understand which features should be prioritised – not abandoned
- Good for communication among stakeholders
  - Helps with prototyping
Nielsen’s definition

- Individual User
- Specific set of computer facilities
- To achieve a specific outcome
- Under specified circumstances
- Over a certain time interval
  - Could be a relative (sequence) or absolute
Who is Nielsen?

- It is impossible to do HCI commercially and not come across Jakob Nielsen
- Essentially an HCI ‘guru’ and loved and hated in equal measure
Generating

- Can be introspective if personas are completely mythical
  - Not good
- Usually achieved through personal experience, interview or observation
  - More on how to do this later
Scenario wording

➢ **Say what** the user wants, but not **how** to do it
  – No interface assumptions
  – Fair comparison of design alternatives

➢ **Very specific**
  – Exactly what the user wants to do
  – Exact information the user wants to enter or extract
Payback

- The goal of all this is to focus on the feature list and prioritise
- Scott Jenson talks a lot about this in “Simplicity Shift”
- He suggests rating each feature of the product (1, 2, 3) for
  - Usefulness to persona
  - Value to the product
Developing Scenarios

- Describe a complete job
  - Preferably involving several features
  - Focusing on information flow

- Evaluate with real users (if available)

- Think about error conditions
Moving to Prototype

How do we get rid of features from the interface?
- Too much detail (Apple printer, microwave power)
- Hide it
  - Bothers people as ‘cool stuff’ is hidden
- Reduce complex questions
  - Error recovery
What is a prototype?

- Cardboard box, storyboard
- Piece of software
- Chunk of plywood (Jeff Hawkin)
- Dummy box
Why Prototype?

- Evaluation and feedback are central to interaction design
- Stakeholders can see, hold, interact with a prototype more easily than a document or a drawing
- Team members can communicate effectively
- You can test out ideas for yourself
- It encourages reflection: very important aspect of design
- Prototypes answer questions, and support designers in choosing between alternatives

- Supports FAST FAIL
What is Prototyped?

• Technical issues
• Work flow, task design
• Screen layouts and information display
• Difficult, controversial, critical areas

NOT just changing skins!
Low Fidelity

- Uses a medium which is unlike the final medium, e.g. paper, cardboard

- Is quick, cheap and easily changed

- Examples:
  - sketches of screens,
  - task sequences, etc
  - ‘Post-it’ notes
  - storyboards
  - ‘Wizard-of-Oz’
Lo-Fi prototype
Wizard of Oz

DOCTOR FUN

Doctor Whizlie's controversial "talking brain in a jar" is revealed for the shameless fraud that it was.
High Fidelity

- Uses materials that you would expect to be in the final product.
- Prototype looks more like the final system than a low-fidelity version.
- For a high-fidelity software prototype common environments include Macromedia Director, Visual Basic, and Smalltalk.
- Danger that users think they have a full system……..see compromises
Compromises

- All prototypes involve compromises
- For software-based prototyping maybe there is a slow response? sketchy icons? limited functionality?
- Two common types of compromise
  - ‘horizontal’: provide a wide range of functions, but with little detail
  - ‘vertical’: provide a lot of detail for only a few functions
- Compromises in prototypes mustn’t be ignored. Product needs engineering
Final system

- how does the prototype relate to the final system?
- **throw-away**
  - via requirements
- **evolutionary**
  - system used to be the prototype
- **incremental**
  - phased implementation
  - 1 increment informs subsequent increments
Realising Prototypes

- Taking the prototypes (or learning from them) and creating a whole
- Quality must be attended to: usability (of course), reliability, robustness, maintainability, integrity, portability, efficiency, etc
- Product must be engineered
Problems

- Is the critical part different in the prototype and the final system
  - Timing?
- Are some systems attributes un-prototypable?
  - Safety-critical applications
- (Evolutionary)
  - Continual change corrupts the structure of the software
    - Higher maintenance
  - Documentation?
- Throw-away?
  - What is thrown away?
Mobile prototypes

- Biggest problem is hardware
  - Pretty standard for PC systems
  - Slower than the emulator
  - Sometimes not possible (FunkyZoom)
  - Use an iPAQ for everything
Handspring
Current approach

- Paper
- Digital Paper
- Vertical Prototype
- Scenario Prototype
- Deployment on PDA
- Big Experiment