Prototyping & UI Software

John Stasko

Spring 2007

This material has been developed by Georgia Tech HCI faculty, and continues to evolve. Contributors include Gregory Abowd, Al Badre, Jim Foley, Elizabeth Mynatt, Jeff Pierce, Colin Potts, Chris Shaw, John Stasko, and Bruce Walker. Permission is granted to use with acknowledgement for non-profit purposes. Last revision: January 2007.

Agenda

- Prototyping
  - Dimensions and terminology
  - Non-computer methods
  - Computer methods
- UI Software
  - Design tools
  - UI toolkits
  - GUI builder tools
- Poster session preview
- Exam preview
Design Artifacts

- How do we express early design ideas?
  - No software coding at this stage

- Key notions
  - Make it fast!!!
  - Allow lots of flexibility for radically different designs
  - Make it cheap
  - Promote valuable feedback

*** Facilitate iterative design and evaluation ***
Dilemma

- You can’t evaluate design until it’s built
  - But...
- After building, changes to the design are difficult
  - Simulate the design, in low-cost manner

Prototyping Dimensions

- 1. Representation
  - How is the design depicted or represented?
  - Can be just textual description or can be visuals and diagrams
- 2. Scope
  - Is is just the interface (mock-up) or does it include some computational component?
Dimensions (contd)

- 3. Executability
  - Can the prototype be “run”?
  - If coding, there will be periods when it can’t

- 4. Maturation
  - What are the stages of the product as it comes along?
    - Revolutionary - Throw out old one
    - Evolutionary - Keep changing previous design

Terminology (1)

- Early prototyping

- Late prototyping
Terminology (2)

• Low-fidelity prototype
  Far from final form of system, such as paper, drawings, etc.

• High-fidelity prototype
  Close to final form of system, much more realistic to actual application

Terminology

• Horizontal prototype
  Very broad, does or shows much of the interface, but does this in a shallow manner

• Vertical prototype
  Fewer features or aspects of the interface simulated, but done in great detail
Rapid Prototyping Methods

- Non-computer vs. computer-based

Typically earlier in process

Typically later in process

Non-Computer Methods

- Goal: Want to express design ideas and get quick & cheap opinions on system

- Methods?
1. Design Description

- Can simply have a textual description of a system design
  - Obvious weakness is that it’s so far from eventual system
  - Doesn’t do a good job representing visual aspects of interface

2. Sketches, Mock-ups

- Paper-based “drawings” of interfaces
- Good for brainstorming
- Focuses people on high-level design notions
- Not so good for illustrating flow and the details
- Quick and cheap -> helpful feedback
Physical Mock-Ups

• Wooden blocks and labels - device control

(Three versions of a hand-held controller)

Physical Mock-Up

• Styrofoam and Buttons

Spring 2004 CS 4750 project “Golf Caddy” by:
Chris Hamilton
Linda Kang
Luigi Montanéz
Ben Tomassetti
3. Storyboard

- What is it?

Storyboarding

- Pencil and paper simulation or walkthrough of system look and functionality
  - Use sequence of diagrams/drawings
  - Show key snap shots
  - Quick & easy

Scenario

The user begins by waking up to the touch-screen display. He can't quite decide what he wants to order. He presses the "Single Item" button. Once there he sees a menu for everything the store has to offer. He decides he'd like a hamburger. He presses the button for Hamburger and continues to the next screen. He then decides he'd also like a Coke and a Fries. After adding these to the running total, he decides the cashier is breaking the window it looks like a truck will be here any minute. He presses the "Place Order" button. After placing the order he is satisfied and presses "Finish Order". He then decides to pay with cash and heads to the checkout. A few minutes later he realizes he forgot his keys and goes on his way.
**Uses / background**

- Very similar in nature to:
  - Comic art / cartoons

- Used in:
  - Movie / multimedia design
  - Product / software development

**Example**

Sketches solve two problems with use of more fully-developed prototypes

- User reluctance to suggest changes to what might look like a finished product
- User focus too much on details (graphic design, etc) of UI rather than big picture
Elements of storyboard

- Graphical depiction of scenarios

- 5 visual elements
  - Number of frames/panes
  - Use of words
  - Level of detail
  - Inclusion of people
  - Time passage

How is it done?

- Novice designers’ process
  - Individual brainstorming about ideas
  - May do some quick initial sketches
  - Team meeting to discuss ideas / drawings
  - Decision on what to draw
  - Spend next ~8 hours together drawing
    - Co-location allows quick feedback
    - Can also glance at what others are drawing for inspiration
How is it done?

- **Expert designers’ process**
  - Get assignment
  - Individual brainstorming about ideas
    - Determine the story
    - Includes a lot of sketches using pencil + paper
    - A very iterative process through a lot of initial drafts
  - Team meeting to discuss ideas / drawings
    - Share copies of drawings
    - Discuss what stories should be told
  - Repeat
  - Generate more polished art for presentation
  - Develop

Experts’ advice on storyboarding

- Keep it short: 1 interaction/activity per storyboard
- More is not always better. Why?
  - May lose focus of story
  - May lose reader’s attention
- Biggest challenge? Experts say:
  - Must be able to succinctly tell story
Keep the drawing short

- Drawing more is not always needed...

Use taglines / captions

- Keep it short
Inclusion of actors and objects helps to create empathy

- "The first thing users will want to know is why do I even care about this application?"
- Can show how the user interacts with the system and how the system affects the user

When to show time passing

- Time passing is implicit
- Only needed when gross changes or minute changes need to be explicit
- Readers bring own expectations of how much time passes into the storyboard
Some advice

- Figure out your story
- Identify main points in the story
- Draw 3-5 frames/panes (to match the main points)
- Keep it simple...
- Add taglines / text to enhance understanding
- Pilot storyboards & iterate

4. Scenarios (aka Use Cases)

- Hypothetical or fictional situations of use
  - Typically involving some person, event, situation and environment
  - Provide context of operation
  - Often in narrative form, but can also be sketches or even videos
Scenario Utility

- Engaging and interesting
- Allows designer to look at problem from another person’s point of view
- Facilitates feedback and opinions
- Can be very futuristic and creative
- Can involve social and interpersonal aspects of the task

5. Other Techniques

- Tutorials & Manuals
  - Maybe write them out ahead of time to flesh out functionality
  - Forces designer to be explicit about decisions
  - Putting it on paper is valuable
Computer-Supported Methods

- Simulate more of system functionality
  - Usually just some features or aspects
  - Can focus on more of details
  - Typically engaging
  - Can lead to “stale” design, can focus user (or customer) too much on the details of the interface, too early in the design process
  - Danger: Users are more reluctant to suggest changes once they see more realistic prototype

Prototyping Tools

- 1. Draw/Paint programs
  - Draw each screen, good for look

Thin, horizontal prototype
PhotoShop, Corel Draw,...
Through international education, experience and community service, United World Colleges enable young people to become responsible citizens, politically and environmentally aware, and committed to the ideals of peace and justice, understanding and cooperation, and the implementation of these ideals through action and personal example.

Photoshop

Illustrator
Prototyping Tools

- 2. Scripted simulations/slide shows
  - Put storyboard-like views down with (animated) transitions between them
  - Can give user very specific script to follow
  - Often called chauffeured prototyping

Examples: PowerPoint, Hypercard, Macromedia Director, HTML

Powerpoint Transition Controls

Mouse click actions:
- Next slide
- Previous slide
- First slide
- Last slide
- Last slide viewed
- End show
- Custom show
- URL
- File
Scripting Example

But Beware!

By Scott Adams

DILBERT

AS YOU CAN CLEARLY SEE IN SLIDE 397....

GAAAAH!

"POWERTPOINT" POISONING.
Dreamweaver

Fireworks
Apple Hypercard

- Once a very popular prototyping tool for simulating UI
- Allows control of simple card transitions
- More complex behaviors

```plaintext
on mouseUp
  play "boing"  
  wait for 3 seconds
  visual effect wipe left very fast to black
  click at 150,100
  type "goodbye"
end mouseUp
```

Macromedia Director

- Combines various media with script written in Lingo language
- Concerned with place and time
  - Objects positioned in space on “stage”
  - Objects positioned in time on “score”
- Easy to transition between screens
- Can export as executable or as Web Shockwave file
Prototyping Tools

3. Interface Builders

- Tools for laying out windows, controls, etc. of interface
  - Have build and test modes that are good for exhibiting look and feel
  - Generate code to which back-end functionality can be added through programming

- Examples: Visual Basic, Delphi, UIMX, ...

Visual Basic

More to come later today

UI Controls

Control properties

Design area
Flash - A category of its own

True Programming

- Less useful for rapid prototyping, but can save re-coding time down the road
- More constrained in look and feel
- Constrained to traditional interaction styles and methods
  - Hard to think outside the box
- More to come in a few minutes...
Other Prototyping Tools

- Denim

http://gui.r.berkeley.edu

Audio Interface (Telephony) Builder Tools

- SUEDE - Flow-chart for speech interface
  - Landay et al, UC Berkeley (now U Washington)
- Used for wizard-of-Oz studies
- Could be used to drive real system
- gui.r.berkeley.edu/projects/suede/index.shtml
Prototyping Technique

- **Wizard of Oz** - Person simulates and controls system from "behind the scenes"
  - Use mock interface and interact with users
  - Good for simulating system that would be difficult to build

Can be either computer-based or not
Wizard of Oz

• Method:
  – Behavior should be algorithmic
  – Good for voice recognition systems

• Advantages:
  – Allows designer to immerse oneself in situation
  – See how people respond, how specify tasks

Prototyping Tools

• Good features
  – Easy to develop & modify screens
  – Supports type of interface you are developing
  – Supports variety I/O devices
  – Easy to link screens and modify links
  – Allows calling external procedures & program
  – Allows importing text, graphics, other media
  – Easy to learn and use
  – Good support from vendor
Prototyping Summary

- Tradeoffs of simplicity, manageability
- Veracity
- Interactiveness
- Up-front costs vs. down the road costs

- Key: Don’t let the prototyping environment drive or constrain your creativity!!
## Tutorials

**Photoshop/Illustrator:**
- [http://www.planetphotoshop.com/tutorials.html](http://www.planetphotoshop.com/tutorials.html)
- [http://thetechnozone.com/bbyc/Illustrator.htm](http://thetechnozone.com/bbyc/Illustrator.htm)
- [http://studio.pinnacle-elite.com/tutorials/aitut01.html](http://studio.pinnacle-elite.com/tutorials/aitut01.html)

**Dreamweaver/HTML:**

**Fireworks:**

**VB:**
- [http://www.vbtutor.net/vbtutor.html](http://www.vbtutor.net/vbtutor.html)
- [http://webspace.dialnet.com/paul_pbooms/vb/tutor.html](http://webspace.dialnet.com/paul_pbooms/vb/tutor.html)

**Flash:**
- [http://www.uic.edu/depts/accc/seminars/flashintro/index.html](http://www.uic.edu/depts/accc/seminars/flashintro/index.html)

**Director:**
- [http://www.herts.ac.uk/lis/mmedia/directortutorial/](http://www.herts.ac.uk/lis/mmedia/directortutorial/)

## UI Software & Programming

- **OK, let’s return to what we were talking about earlier**

- **The final level up...building the actual application**
User Interface Software

- What support is provided for building graphical user interfaces?
  - UI toolkits
  - GUI builder tools

- Let’s examine some background...

GUI System Architecture

What does it look like?
Layered Architecture

- Application
  - Higher level Tool
  - UI Toolkit
  - Window Manager
  - Window System
  - OS
  - Hardware

Window System

- Allocates and manages
  - Regions of display to application programs, keeps programs out of each other's way
  - Input devices (keyboard, mouse) to application, routes input events

- Called by application program to
  - Create/delete windows (ie, allocate resource)
  - Operate on windows (move, resize, bring to top, hide, give name, clear)
  - Enable input devices
  - Get input from user via interaction devices
Key Windowing Concepts

- Server/client
- Window manager ≠ window system
- Virtual display/input device abstraction
- Window hierarchy
- Event notification: queue vs. callback
- User actions
- Input focus
- Consequences of server actions (redraw)

Client-Server Window System

(After Fig 10.2, Dix, Finlay, Abowd and Beale)
Client- Server

- "Policy-free" server, not seen by user
- A Window manager client gives the "look and feel", not the window system (server)
- Multiple clients, each seen by user on terminal
- Linux/Unix WMs built on the X Window System include
  - LVWM - Open Look Virtual Window Manager
  - FVWM - Feeble Virtual Window Manager
  - MWM - Motif Window Manager
  - AfterStep
  - GWM - Generic Window Manager

Window Hierarchy
Event Notification

- Event types
  - User actions - mouse movements/button clicks, keyboarding, enter or leave window
  - Server actions, such as making window visible - client may need to redraw window
- Event queue – program examines the queue, calls an action routines, determined by event type and screen object under cursor
- Callback – the window system notifier is told which action routine to call for each type of event for each object on screen

Input Focus

- To which client do events go?
- Not always where the cursor is located
- Dynamic dragging outside of window
- Type into one window while "mouse" in another
Separation of Concerns

- Application
  - Core functionality
  - Operations
  - Data

- Interface
  - Interface components
  - Graphics
  - I/O

Should these be separated architecturally and in code?

Why or why not?

How Does a Toolkit Work?

- What exactly does it provide?
- How is it organized?
**Toolkit Workings**

- User takes actions, interacts with interface
- Those actions must be delivered to application in meaningful ways
- Application takes appropriate actions, perhaps updating display

**Seeheim Model**

**Conversational model**

Dominant model for long time
Object Model

- UI is collection of interactor objects (often called widgets)
- User directly manipulates them
- Objects responsible for transmitting user actions to application in meaningful ways

Model-View-Controller (MVC)

- Developed for Smalltalk (Alan Kay)
- A refined object model: V+C = UI
- Used in JAVA’s Swing UI widget library

State and behavior of Model
Create and update a View of the model
Control/manage user interaction with the model
MVC Example - a Push Button

- Model - a boolean - on or off
- View - a drawing - in each possible state
- Controller - tell model to change state, and view to change view
- Note - most buttons have more complex behavior than this


MVC Flexibility

- Clear separation of concerns makes changes easy
  - Want a different button appearance? Change the view, nothing else.
  - Want mouse_over rather than mouse_down to change state?

Locus of Control

- "Traditional" software
  - Control is in system, query user when input needed

- Event-driven software
  - Control is with user (wait for action)
  - Code reacts to user actions
  - More difficult to write code this way, harder to modularize, etc.

Classical Event-Driven Program

![Diagram of classical event-driven program]

- U I
- System
- Entry point
- What's that?
- Main application code
**UI Toolkit**

- What application programmer typically programs with
- Combination of interface objects and management behaviors
- Usually object-oriented now
- Library of software components and routines that programmer puts together
  - X Windows: X Toolkit & Motif
  - Macintosh: Mac Toolbox, MacApp
  - Windows: Windows Developers’ Toolkit
  - Java: Swing

---

**Classical Approach to Input Event Dispatching – Not so Good**

![Diagram showing classical approach to input event dispatching]

- Application program polls the event queue
- Examine events, call processing module
- Process event type 1
- Process event type 2
- Process event type 3
Classical Event-Driven Program

- Initialize display & system
- Repeat
  - Wait for and get next user action
  - Decipher action
  - Take appropriate action
  - Update display
- Until Done

Callback Routine – Way to go

- Software procedure, part of application
- Invoked when particular action occurs to UI component, such as pressing a PushButton
- Procedure is invoked with event parameters
Window Approach to Input Event Dispatching - Good

Examine events, call processing module

- Callback to process event type 1
- Callback to process event type 2
- Callback to process event type 3

Application program

Window Manager Notifier

Application program callbacks invoked by window manager

Widgets are Source of Events

- Widgets are typically structured as objects
- JAVA’s Swing uses MVC model
  - Each widget has three parts
Windows Widgets

- Buttons (several types)

- Scroll bars and sliders

- Pulldown menus

More Windows Widgets

- Palettes

- Dialog boxes

- Windows and many more...
Example – X & Motif

- Object-oriented hierarchy of UI interactors called widgets
  - Associate callback routines in your code with them
- Interface is built up by putting child widgets “onto” parent widgets
**Widget Hierarchy**

- Widgets organized into inheritance hierarchy

```
Primitive
  ├── Text
  │     ├── Push Button
  │     └── Drawn Button
  └── Label
     └── Toggle Button
  ├── Button
  └── Scroll Bar
```

**Widget**

- Visual appearance
- Set of tailorable attributes

```c
PushButton {
    Color Background;
    int MarginLeft;
    int MarginRight;
    int BorderWidth;
   Pixmap ArmPixmap;
    Boolean FillOnArm;
    CallbackList ActivateCallback;
}
```

- Interactive behavior
Widget Use

- Set up widget attributes
- Create widget object (as child of parent widget)
- Define callback or event procedure for widget

Callbacks associated with objects and events
Multiple Callbacks per Object

- Example - button object with 5 callbacks
  - Mouse enter - (1) highlight
  - Mouse button down - (2) additional highlighting
  - Mouse leave while button down - (3) remove highlight
  - Mouse button up - (4) remove highlight, (5) perform action

Widget and Callback

```
n = 0;
xmstr = XmStringCreate("Color", XmSTRING_DEFAULT_CHARSET);
XtSetArg(args[n], XmNlabelString, xmstr); n++;
XtSetArg(args[n], XmNbackground, red); n++;
colorbut = XtCreateManagedWidget("colorbutton",
    xmPushButtonWidgetClass, focusrowcol, args, n);
XtAddCallback(colorbut, XmNactivateCallback, colorChangeCB, id);

void
colorChangeCB(Widget w, XtPointer userdata, XtPointer evtdata)
{
    // Actions
}
```
Main Program Event Loop

```c
void CheckXEvents()
{
    XEvent xev;

    while (XtAppPending(_context)) {
        XtAppNextEvent(_context, &xev);
        XtDispatchEvent(&xev);
    }
}
```

OO Systems

- Java’s GUI programming done with AWT and Swing
- More distributed model (separate threads)
- Primary action here is dispatching events to objects (widgets) as messages
- Delegation important
  - Can make particular objects responsible for event handling
Example - Java AWT

```java
public void mouseReleased(MouseEvent e){
    System.out.println("Changing color");
    if (bHighlight)
        frame.setBackground(highlight);
    else
        frame.setBackground(normal);
    bHighlight = !bHighlight;
}
```
Java Output

Get What You Ask For

6750-Spr '07 102
Higher Level Tools

- Provide assistance or some automation in developing UIs

- Four types
  - Language - high-level programming language
  - Application framework - for specific application domain
  - Model-based systems - driven by UML or DB Schema
  - Interactive GUI Builders - by far the most prevalent and accessible

GUI Builder Tools

- Why build graphical (visual) interface with textual commands?

- Why not show what you want it to look like?

Tool Methods

- Work area (interface being built)
- Drag and drop interactors/widgets onto work area
- Specify position, color, look, etc.
- Often provide Build/Test modes

Example: dtbuilder (Motif)
Example: Visual Basic

UI Controls
Design area
Control properties

Widgets in VB Toolbox

Pointers
TextBox
Label
ComboBox
Frame
Option Button
Check Box
Command Button
Combo Box
Horizontal Scroll
List Box
Vertical Scroll
Timer
Text Box
Directory List Box
Shape
Data
Image
Line
OLE
Connecting Code to Widgets

Making Menus
Interested in This?

- Take CS 6456, Principles of UI Software
- Should have a good programming background

MidTerm Exam

- Next Tuesday
- Short answer style questions
  - Know your definitions, terms, concepts
  - Material from lecture & book
Poster Session

- Thursday during class
- Buy a poster board (bookstore)
- Display your design ideas
  - Lots of pictures
  - Explanatory text
- Get some good feedback

Upcoming

- **Mid-term Exam**
- Poster Session
- Dialog styles