Interaction Design

IMS2000
Interface and Interaction Design
Dan Eaves
August, 2004

A Digression

• Do not confuse
  – A standard GUI system with
  – An interactive Web site

• Contemporary Web sites are
  – Hard to program but trivial in functionality
  – Have their own design conventions
    • Hypertext jumps
    • Minimal programmatic functionality

How Can I Say This...

• In computing, new (and maybe marginal) application areas get most of the attention (especially in the ivory tower)
  – They are new and different.
  – They aren’t boring.
  – The constraints of the domain aren’t known and do need to be explored.
  – They are where the marginal $s are going

• It doesn’t make them particularly important...
• (Web-based programming is the flavour of the week.)

Microsoft Has Decided...

• Most future apps will be Web-centric.
• They will follow the Web/HTML model of transparently distributed components and resources.
• They will of necessity follow (mostly) the conventions of HTML files with Dynamic HTML and mostly of server-side processing, with XML structured data as the medium of exchange.
• The people who brought you dickless (whoops, diskless, aka thin) desktop computing invented J2EE, which Microsoft is slavishly reinventing.
• The success of the model is still in doubt. (But it is where the money is right now! And why VB.NET is such a bitch to learn.)

We’ll Focus on Traditional Desktop GUI Apps
(And leave Web apps to the true believers)

• The First Commandment:
  Don’t design systems for yourself!

• The First Corollary
  The First Commandment Isn’t Easy

• So you have to catch some users?
It’s Not that Easy

• For the rest of the lecture we’ll be working through something called scenario based design:
  – Primary sources
    • Alan Cooper and Robert Reimann’s *About Face 2.0: The Essentials of Interaction Design*. (Wiley 2003)
  – Both are highly practical and very readable

The Jargon: Things to help us focus

• **Personae**
  – The name of those happy and sad masks you see in theatres
  – Detailed *personifications* of particular users of your system

• **Scenarios**
  – *Tasks* (from sitting down to getting up) which a persona will accomplish with your system
  – Tasks have particular goals
  – The system must make the scenarios easy and logical

Interface Versus Interaction

• Consider yourself designing/building your system
  • You:
    – Put things on the screen
    – Move them around
    – Stop when they look right
  • The screen is a *static thing* of many pieces in the process of being built.

• Consider someone using your system
  • The screen is seen as a *dynamic whole* (chunk) in the process of being used to perform a task.

• You are a designer of interactions, not a builder of interfaces
  – (Even though it doesn’t feel that way!)

Now start designing the interaction.

• Note:
  – the change to the first tab
  • No need to remember who we are working on
  – What’s big and bold
    • The data input is always more important to the user than the prompts
    – Each chunk is in a frame with a bold heading.

Now let’s look at the margins
• Details are hidden until necessary
  – (Never use pop-up windows for extra information.)
  – (Well, hardly ever, and when you do they must be modal... like an initial passworded log-on)

• But this is easy to get wrong:
  – Gaaakkkkkk.
  – How might we fix this?????

What’s That Ugly Yellow???
• I want to distinguish between parts of the screen
  – "owned" by the program, or
  – "owned" by the user.
  – Anything in that shade of yellow, in any system I design, is owned by the program, and locked.
    • (I really wish I’d picked a different colour back in 1995!)

• I never tell the user this. They wouldn’t remember if I did. But they learn by experimentation.

Ownership
• The important part of the screen is owned by the user.
• This is where they put data, look at results, etc.
• The other parts of the screen are owned by your program.
  – Not as important
  – If the system is used regularly, the user will quickly learn to ignore this part of the screen
  – Data solely for program purposes (e.g., magic numbers) should be invisible to the users!
• Your design must reflect the relative importance of the two areas.

Dialog Style
• In a dialog,
  – you talk and then
  – someone else talks.
• Your system should let the user do the talking until she passes control to your program.
• Do not interrupt your user.
  – Let the system remain in an error state until the user indicates the task is completed.
  – Use a status bar, never pop-ups.
  – Never, ever, ever use masked edit controls

Scenario-Based Design
• As a critical part of systems analysis:
  – Watch the users walk through a standard set of tasks.
  – Have them talk their way through each process.
    • (Keep quiet, don’t disturb them, questions can come later.)
  – Learn the users’ language, noting down what they call both elements and tasks.
  – Do this with a number of users, to avoid particular user’s idiosyncrasies.
  – Write narrative descriptions of each task, using their vocabulary.
  – Review these scenarios with users, asking for corrections.
    • Only then start asking, “Why?”
• Keeping your mouth shut is hard!
A Real Case

• The Context
  – When developing residential projects
  • Many of the buyers are small scale investors
  • Most sales are “off the plan”, made before building is even started: Greatly reduced stamp duty
  • The developer uses the sales to fund the actual development
  – The system has to
    • Be customised to particular developments
    • Be useful to the financially naive
    • Be capable of including many multimedia add on sales materials
    • Be capable of being badged for particular developers

My Personae

• This financial system was designed for two public personae and two hidden ones:
  – Adrian the real estate agent. (public)
  – Brian Penny, a possible investor with little or no experience in investing. (public)
  – Brian Penny’s brother-in-law the accountant. (private)
  – The property developer, releasing a multi-million dollar development hoping to attract investors to “buy off the plan”. (private)

Public-Private?

• Public = people who directly use the system
• Private = people involved who don’t directly use it.

• You have to take both types into consideration when you design your system!

(The Value Chain)

• My company sells a customised version of my standard system to
• The property developer who gives it on CD to
• Many in house and external real estate sales staff who use it to sell
• Individual units to investors buying off-the-plan

• (My company may also produce custom multimedia presentations integrated to my software, as well as web sites, etc.)

Brian Penny, the Buyer

• The most important personae
  – 40 year old sergeant in Victorian Police, very secure job, not rich, married, 2 kids
  – Worried about quality of life in retirement
  – Inexperienced investor, highly risk averse, very worried about implications of investment
  – Wants to Get Rich Slowly.
  – Doesn’t trust real estate salesmen
  – Will take time to make decision
  – Uses computer systems, but doesn’t like them particularly
  – Will only use my system during decision-making process

Adrian the real estate agent

• What can I say?
  – Is a real estate salesman……
  – Doesn’t usually sell investment properties, esp. not “off-the-plan”, so needs lots of help.
  – Proud of non-existent computer skills
  – Has to be made to look competent at computer system.
  – Will use the system a lot for a couple of months.
Brian and Adrian
- Want a computer system that’s:
  - Very easy to use
  - Asks for minimum details
  - Quickly provides answers
  - Shows the implications of the investment mid- and long-term
  - Allows experimentation
  - Provides output which can be taken away

Brian’s Brother-in-law the Accountant, 1
- Brian doesn’t trust Adrian (surprise).
- Brian knows he’s not an expert investor.
- But Brian has a (friend, relative, workmate) who is an expert and who Brian trusts.
- The system has to sell the investment 2nd hand to the brother-in-law (BiL).
- BiL will go over the documents, ask lots of questions, and prime Brian to do a lot of exploring…

Brian’s Brother-in-law the Accountant, 2
- Brian goes back to Adrian with the BiL’s questions.
- The system has to be able to answer all of them, allowing Brian to change all of the assumptions hidden in the bowels of its calculations.
- There are lots of assumptions Brian can look at, play with, change in his 2nd and subsequent system uses. Indeed, this is the most important phase of system use, when the system proves its credibility.
- Brian can trust the system even if he doesn’t trust Adrian! The system requires minimal input (simple in appearance) but includes all possible variables, which are available to check and change (complicated in reality)

The Property Developer
- Wants “sizzle”
- Will (maybe) have the accountants check out the logic
- Sees the cost of the system as trivial in the context of the cost of the development
- Pays us!