



fatronik  
tecnalia



# User Centred Design Part 1

## Task based design

Stefan Carmien  
June 16th 2009  
Fatronik-Technalia  
Salon de Cuersos



# Diffusion of innovations

Rogers, the father of adoption studies

- Diffusion of Innovations (1962-2003)
- Looking at how a new technology is adopted
- Sociological approach (i.e. the whole picture)

# Why do we care?

## The Horrible Truth

- 1/3 of all AT is abandoned in first month of ownership
- Abandonment is higher as technology gets more complex
- Abandonment is higher as application moves from sensory to cognitive

## Even worse

- Abandonment has been found to be as high as 70% and often is above 50%

# People that adopt technology

- **Innovators:.**
  - disposable income .
  - technical knowledge
  - Patience and motivation
- **Early Adopters**
  - High status
  - Look to innovators for guidance
- **Early Majority**
  - Between the very early and the relatively late to adopt makes them an important link in process
- **Late Majority**
  - Need peer pressure to adopt
- **Laggards**
  - Reluctant to change (economics)
  - Resistant to change (social structure)

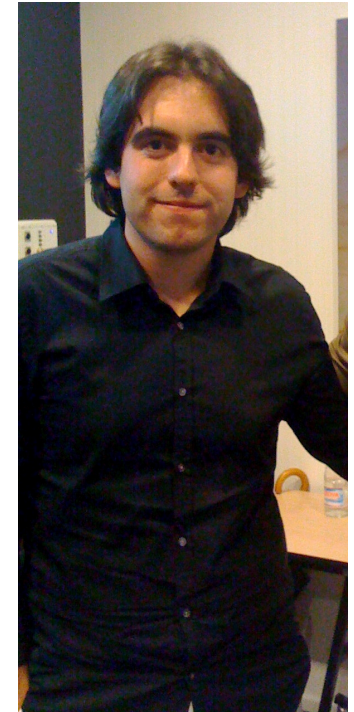
# Who are our customers?

Us? Them?

Who is us? - depends on what we are doing...

Who is them? Depends on what the product is for

Who is paying for it?





# Perceived attributes of innovations

- **Relative Advantage**
  - Is this better than what I have?
- **Compatibility**
  - Will this work with what I have (what I already know?)
- **Complexity**
  - How hard is this to learn?
- **Trialability**
  - Can I just try this or do I have to throw away the old thing before I even try it?
- **Observability**
  - If the neighbours see the new thing, it's easier for them to adopt




# Task Based User Interface Design

“Task based” ? What about user centred design?

User + environment + goal = task

So this is really ‘User plus’ centred design

This is only about part of the design process

- Functionality can be specified and evaluated
  - User interfaces are not so bounded
  - How to make a UI that is ‘good’ is not trivial
- 





# Task Based User Interface Design

We will be using :

*Task-Centered User Interface Design*

*A Practical Introduction*

by Clayton Lewis and John Rieman

Available here:<http://hcibib.org/tcuid/index.html>

Get it - a useful guide and check list for any project



# Task Based User Interface Design

- Figure out who's going to use the system to do what
- Choose representative tasks for task-centered design
- Plagiarize
- Rough out a design
- Think about it (evaluate it without users before you build it)
- Create a mock-up or prototype
- Test it with users
- Iterate
- Build it
- Track it
- Change it



# Figure out who's going to use the system to do what

Is this something a user wants to do or just something you *can* do?

Find real people who are potential users and talk to them

- Lots of things seem wonderful to designers but leave real users bewildered as to why they should use it
- Build it and they WON'T come

# Choose representative tasks

Now put your user to work!

Create some tasks that she will need to do with the new system

They need to clearly say what needs to be done but \*not\* how to do it

Task qualities:

- It is very specific
- It describes a complete job
- Tasks should say who user are

Comment: the user is not always right (especially about novel technology)

Comment 2 : if a system is supposed to be good for everybody you better be sure it's good for SOMEBODY



# Plagiarize

“Good artists copy; great artists steal”

## Intelligent borrowing

Leverage:

- Good design
- Existing user skills

Don't just copy widgets, copy interaction

Key to good borrowing is knowing WHY things are done

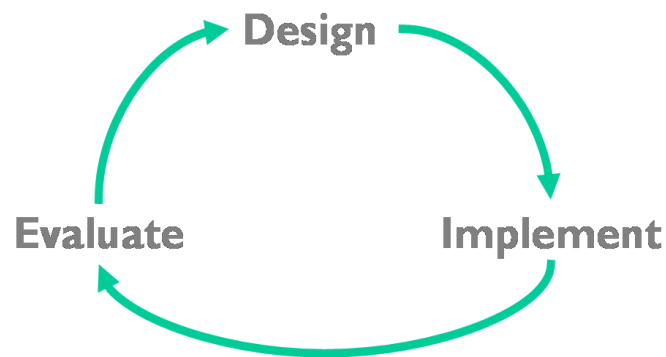
- Fitts law (motion and size)
- Memory (recognise vs.. recall)

# Rough out a design

This means *ROUGH* design

It would be good to do it only once but this will be iterative  
Paper prototypes work fine at this stage

You want something exterior and specific to work with



IBM Bullseye Masthead >>

Internet Quoting

Overview

My Request Gallery

Public Requests

Supplier Profile

Feedback

Help

My Request Gallery

Search on:  Search for:

Click on:

- a radio button in the left-most column to select a bid; then a button below to act on that bid.
- column heading to sort records by that field (or "twisted" if Domino used)

<input type="radio"/>	Request Name	Status	Buyer	Post Date	Due Date
<input checked="" type="radio"/>					
<input type="radio"/>					
<input type="radio"/>					
<input type="radio"/>					
<input type="radio"/>					
<input type="radio"/>					
<input type="radio"/>					
<input type="radio"/>					

White text on blue background

Launches new browser window -> "Print/Export Request"

Slide 15 of 26



# Think about it

Evaluating a design before you have users use it

- Cognitive walkthrough
- Action analysis
- Heuristic evaluation

Users time is not free

Iterations between users sessions can be expensive



Comment: Proxy users for initial design feedback

# Cognitive walkthrough

A formal way of imagining people using the system

How to do it:

- Have a user
- Have a task
- Have a prototype
- Make a believable story out of each action
  - Where you can't do this you may have a design problem

Comment: This is a design tool not an evaluation tool - problems are *good* things



# Action analysis

At it's most rigorous this is keystroke level quantitative analysis of tasks.

- Lots of work for even small task,
  - For each mouse/key/cognitive task...
  - Recognition time+ reaction time+ action time....
- This is only useful for tasks that are performed  $\sim 10^4+$  times a day
- Better for our research is 'back of envelope approach'
  - List the actions (granularity is critical)
  - Count'em up
  - Assume each one takes 2-3 sec

# Heuristic evaluation

## Heuristics?

- Good old guidelines
- Where to get them

## Who does this?

- Your buddies
- Real experts
- Proxies

Multiple evaluators  $\Rightarrow$  more potential problems caught

## Drawbacks:

- Task coverage problems
- Misses cross task interaction



# Heuristic evaluation

## Nielsen and Molich's Nine Heuristics

1. Simple and natural dialog
2. Speak the user's language
3. Minimize user memory load
4. Be consistent.
5. Provide feedback
6. Provide clearly marked exits
7. Provide shortcuts
8. Good error messages
9. Prevent errors

Comment: Undo is a good thing




# Create a mock-up or prototype

- This should be actual code of some sort
- It should actually interact with the user with the same affordances that the final design will have
- WOZ is OK at this level - you can simulate the functionality
- You don't have to prototype everything (I.e. help screens) you can fill in the rest with paper or WOZ



# Test it with users

- You need to use realistic users  
Your friends are not your friends
  - Ethics issues
  - Test representative tasks  
You can unconsciously choose ‘loaded’ tasks
  - You want to catch failure points  
What can you possibly learn from success?
- 

# Test it with users

Two kinds of information

- Process - what was done
- Bottom-line - metrics

Process is more important

The set-up

Uniform User training & equipment setup

What to record

- Notes (most important)
- Video
- Instrumented PC

# Thinking aloud method

## The technique

- Basic instructions
- Work arounds (i.e. reminders & breakdowns)

## Instructions to user

- Sabotaging yourself with helpfulness
- Sabotaging yourself with task choice/lab setup

## What to do with results

- Notes very important (not just video)
- Take 1/2 - 2 hrs analysis for every hour of testing

## List and categorize breakdowns

## Always do pilot study of usability

*here's where your buddies can be of use*



# Iterate & Build it & Track it ...

- Be willing to throw the first one way
- Keep design rationale notes in code
- Documentation issues (nobody reads the manual!)
- You know the rest....





# Break time

Any questions or discussion before break?

Next we will look at designing with scenarios



# Formal Action Analysis

PHYSICAL MOVEMENTS		
Enter one keystroke on a standard keyboard:	.28 second	Ranges from .07 second for highly skilled typists doing transcription, to .2 second for an average 60-wpm typist, to over 1 second for a bad typist. Random sequences, formulas, and commands take longer than plain text.
Use mouse to point at object on screen	1.5 second	May be slightly lower -- but still at least 1 second -- for a small screen and a menu. Increases with larger screens, smaller objects.
Move hand to pointing device or function key	.3 second	Ranges from .21 second for cursor keys to .36 second for a mouse.
VISUAL PERCEPTION		
Respond to a brief light	.1 second	Varies with intensity, from .05 second for a bright light to .2 second for a dim one.
Recognize a 6-letter word	.34 second	
Move eyes to new location on screen (saccade)	.23 second	
MENTAL ACTIONS		
Retrieve a simple item from long-term memory	1.2 second	A typical item might be a command abbreviation ("dir"). Time is roughly halved if the same item needs to be retrieved again immediately.
Learn a single "step" in a procedure	25 seconds	May be less under some circumstances, but most research shows 10 to 15 seconds as a minimum. None of these figures include the time needed to get started in a training situation.
Execute a mental "step"	.075 second	Ranges from .05 to .1 second, depending on what kind of mental step is being performed.
Choose among methods	1.2 second	Ranges from .06 to at least 1.8 seconds, depending on complexity of factors influencing the decision.