Demos and Prototypes

What is a Software Prototype?

- It depends
- Helps to investigate
    - Technical issues
    - Work flow, task design
    - Screen layouts, information display
    - Difficult, controversial critical areas
    - Match between engineering and customer specification
- Demonstration of proof of concept
- More concrete than a narrative

Why Prototype?

- Enable evaluation and feedback (central to design methodology)
- Improves communication with a team
- Testing ideas out - encourages reflection
- Answer questions!
- Explore alternatives

Prototype Requires Compromises

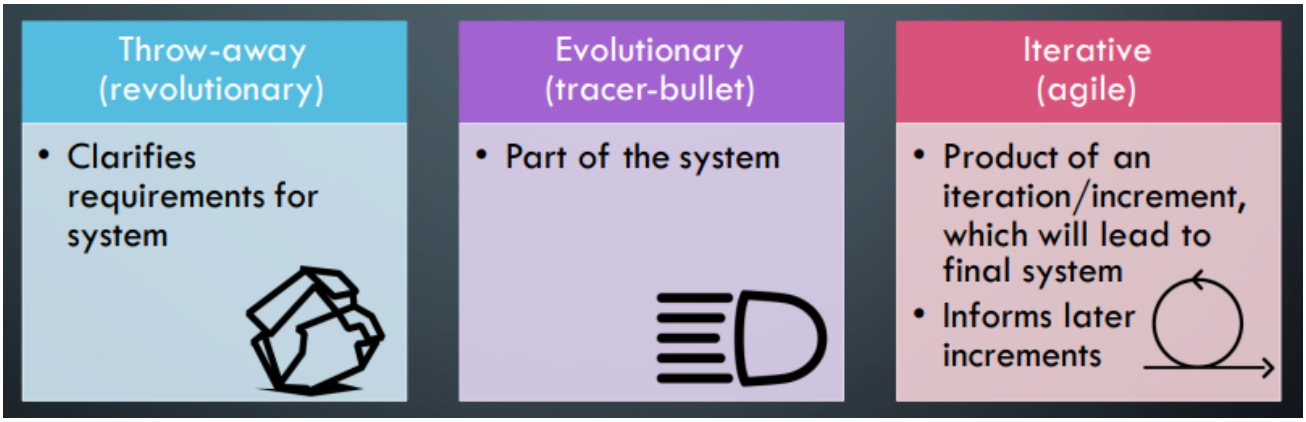
- Slow response, sketchy icons, fake data, limited functionality, limited parameters
- Horizontal Prototype
    - Wide range of functions, very little detail on each e.g.: testing user interface before backend is developed
- Vertical Prototype
    - Provide a lot of detail

Throw-Away Prototype

- Address high-risk issues
    - Uncertainty in requirements
    - User interface design
    - Alternative implementation strategies
    - Technology platform
- Only enough effort to help address specific issues
    - Focus only on the issue, ignore all others
    - No unit tests
    - Too much effort will make you hesitant to throw it away
- Great for trying alternative ideas

Evolutionary Prototypes

- Intended to be early, not necessarily release-able version of the actual software ... will evolve into the final product
- Quality is important (unit test are back)
- Can be put to limited use
- Implementing and validating well-understood requirements -> providing opportunity for change and reorienting if necessary
- Potential Weakness
    - Customers/tester may be hesitant to criticize the underlying problems in something that seems heavily invested/developed



How to Beat Murphy!

1. Make sure your system works
2. Make sure your system works
3. Also, practice and rehearse the demo

How to make sure the system works

- No last minute code changes
- Demo stable version with fewer features rather than an untested version with more
- Rehearse the entire demo
- No, really resist the temptation to tweak the demo
- Test any peripherals

Preparing for the Demo

- List (and test) the tasks you will demonstrate
- Plan what to say
    - Very short overview of the system
    - Explain what the demo will show
- Plan what to say during each test, work out the steps involved
- Ensure that each team member participants and can demonstrate contribution to knowledge

- Rehearse and time the demo
- Think of possible questions - prepare answers

Before the demo

- Arrive Early
- Make sure the system works

During the Demo

- Be prepared to change direction in response to questions
- Allow and encourage the client to try things (if you are prepared for this and it is safe)
- Be prepared to show code or data or documentation
- Be prepared to answer questions

Something Bad Happens

- Stay calm
- Don't make effusive apologies if anything goes wrong
    - Fix the problems quietly
    - Be able to restart the system without recompilation
    - Or have a backup system ready
    - Let one person go on explaining the system
- Don't argue with or contradict team members