Design Process

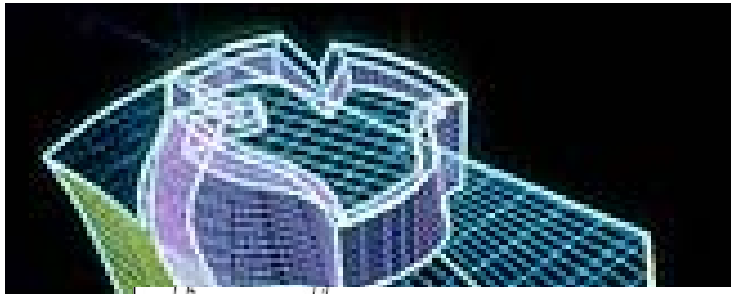
Schematics

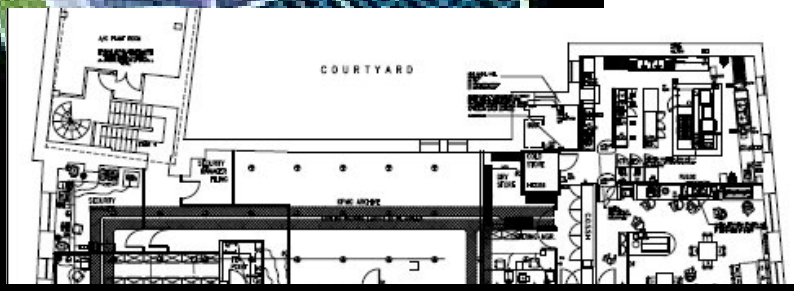
Prototyping

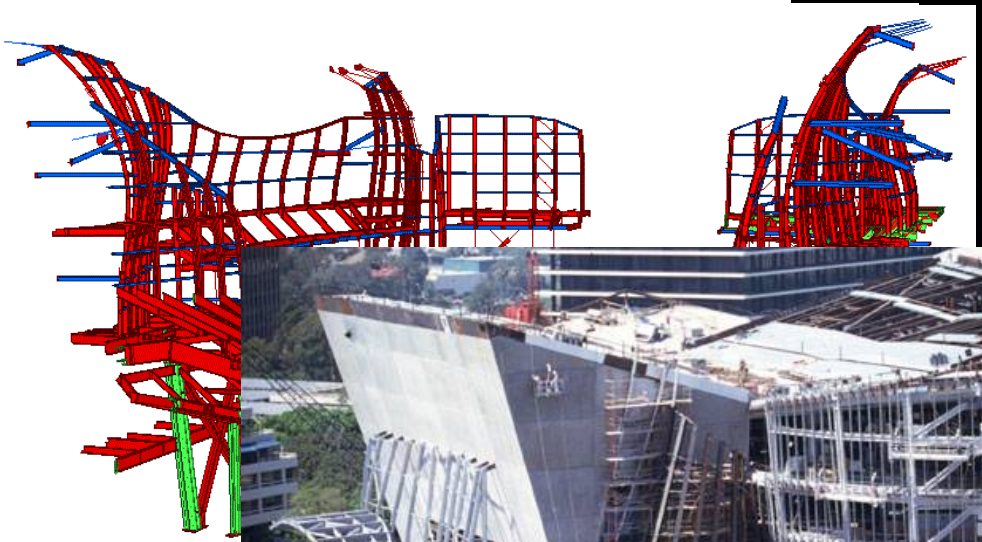
Testing

User Centred Design (UCD)

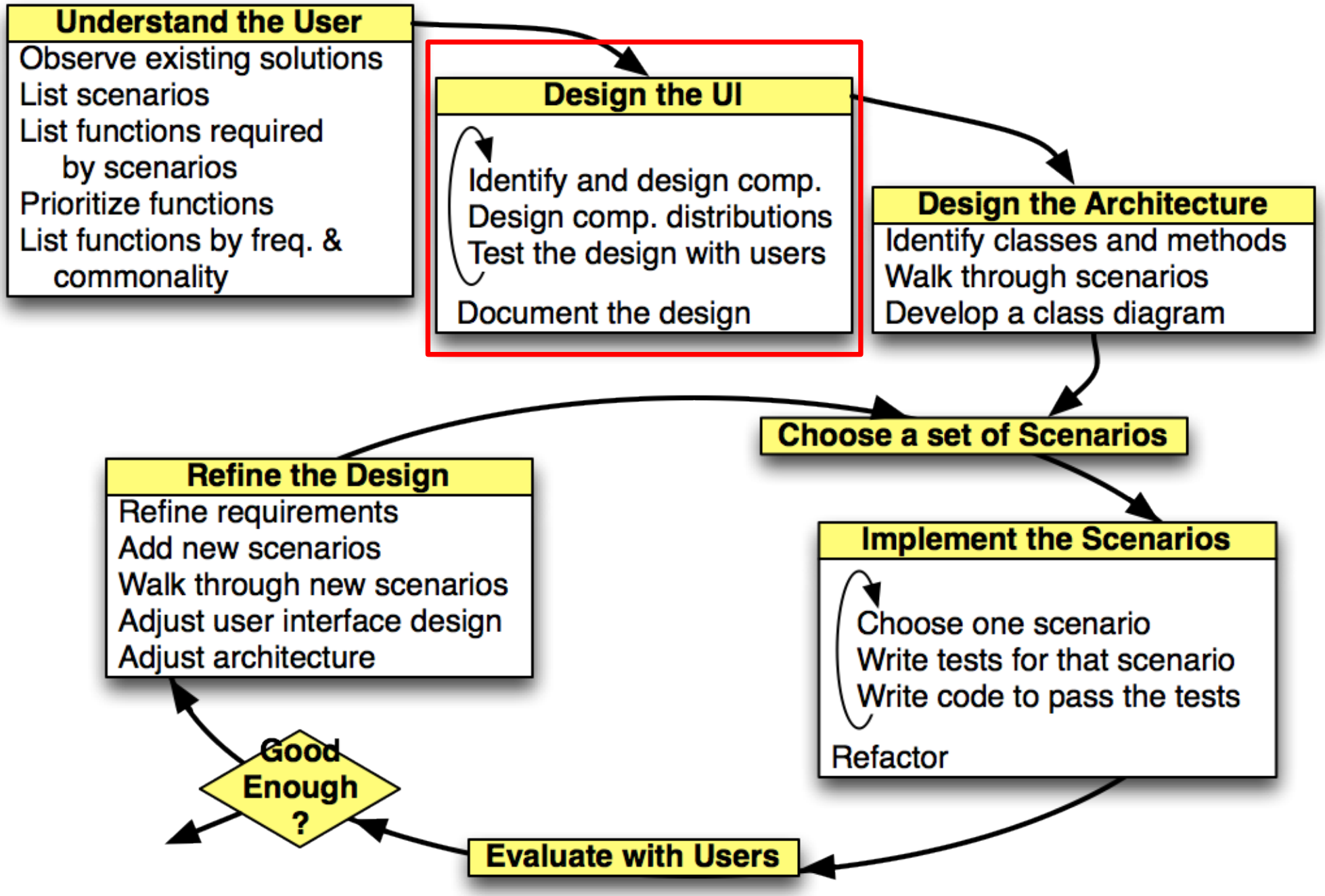
- To make things usable and useful, you need to understand the people who use your software
- UCD is a philosophy:
  - You ask real people about what they need
  - You ask people what problems they have with current solution
  - You think about the people who will use your software
  - You test your ideas with those people

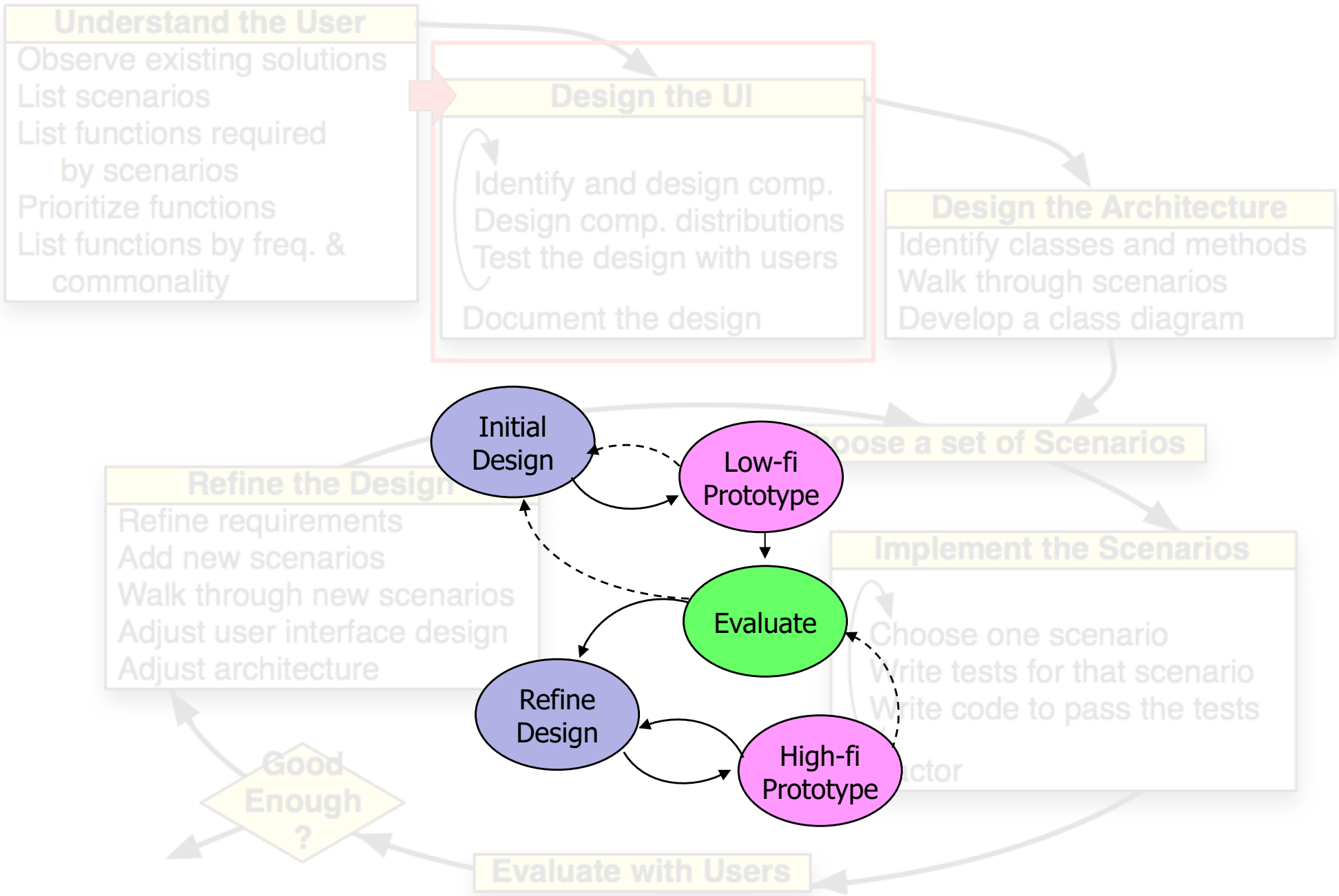






[Frank Gehry, Walt Disney Concert Hall](#)

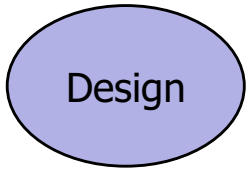

Understand the User
Observe existing solutions
List scenarios
List functions required by scenarios
Prioritize functions
List functions by freq. & commonality
Design the UI
Identify and design comp.
Design comp. distributions
Test the design with users
Document the design
Design the Architecture
Identify classes and methods
Walk through scenarios
Develop a class diagram
Choose a set of Scenarios
Implement the Scenarios
Choose one scenario
Write tests for that scenario
Write code to pass the tests
Refactor
Evaluate with Users
Good Enough ?
Refine the Design
Refine requirements
Add new scenarios
Walk through new scenarios
Adjust user interface design
Adjust architecture


Understand the User
Observe existing solutions
List scenarios
List functions required by scenarios
Prioritize functions
List functions by freq. & commonality
Design the UI
Identify and design comp.
Design comp. distributions
Test the design with users
Document the design
Design the Architecture
Identify classes and methods
Walk through scenarios
Develop a class diagram
Initial Design
Low-fi Prototype
Evaluate
Refine Design
High-fi Prototype
Refine the Design
Refine requirements
Add new scenarios
Walk through new scenarios
Adjust user interface design
Adjust architecture
Implement the Scenarios
Choose one scenario
Write tests for that scenario
Write code to pass the tests
Good Enough ?
Evaluate with Users

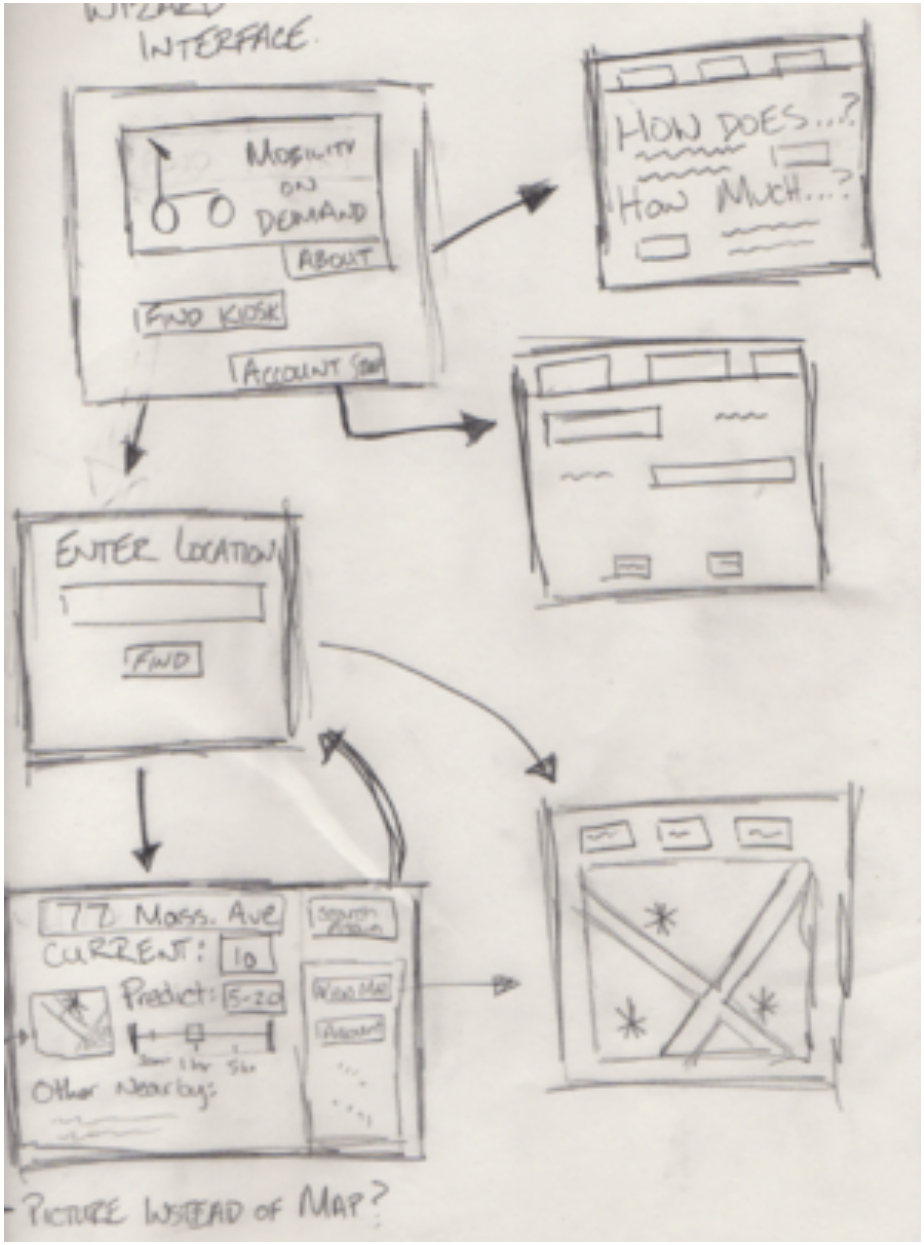
UCD Principles

- **Understand users' needs**: build a product that meets real needs rather than building it because it can be built.
- **Design the UI first**: design the UI first, and then design the architecture to support that UI
- **Iterate**: You won't create the best UI design on the first try; a great design requires iteration
- **Use it yourself**: you'll find obvious problems that can be fixed while it's still easy to fix
- **Observe others using it**: it's critical to observe other people using your UI in a realistic way very early in the development cycle

Design

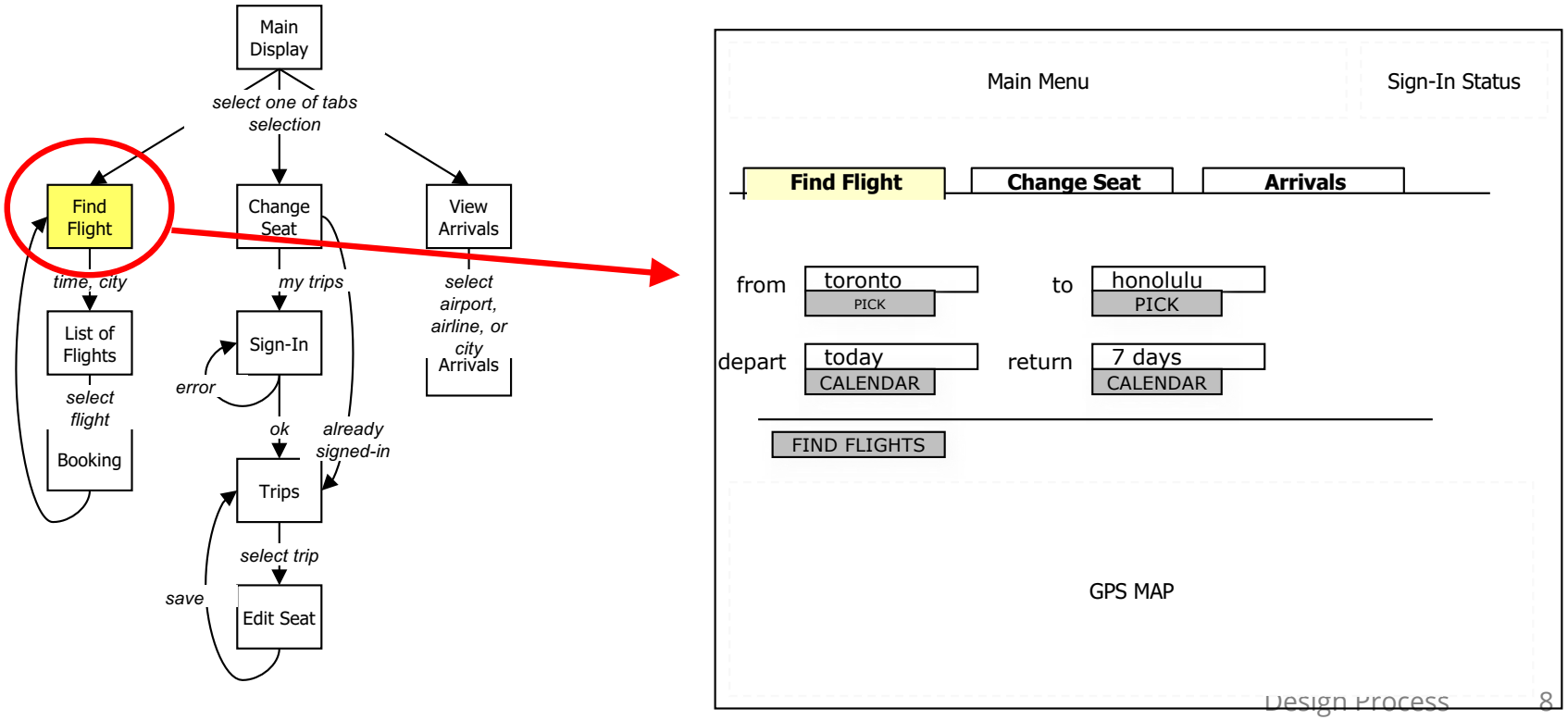


- Temporal aspects of UI:
  - When things appear, the flow from one interface to another
  - "Interaction Sequence Diagrams"
- Spatial aspects of UI:
  - Where things appear on an individual interface section
  - "Interface Schematics"



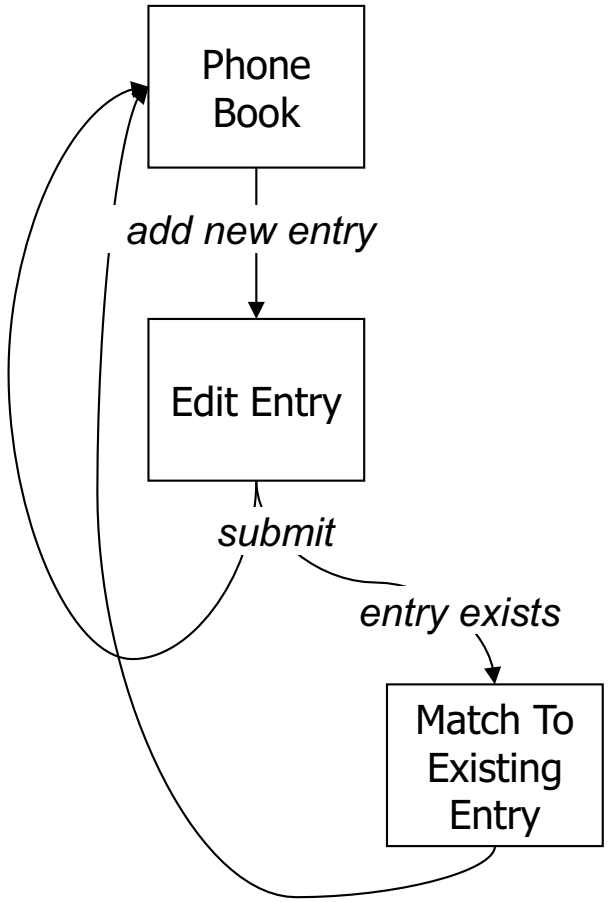
Macro and Micro Structure

- **Interaction Sequences**: macro-structure, convey the "big picture“ of system interaction.
- **Interface Schematics**: micro-structure, convey essential content, and functionality at individual steps of interaction.
- Need both to describe a user interface



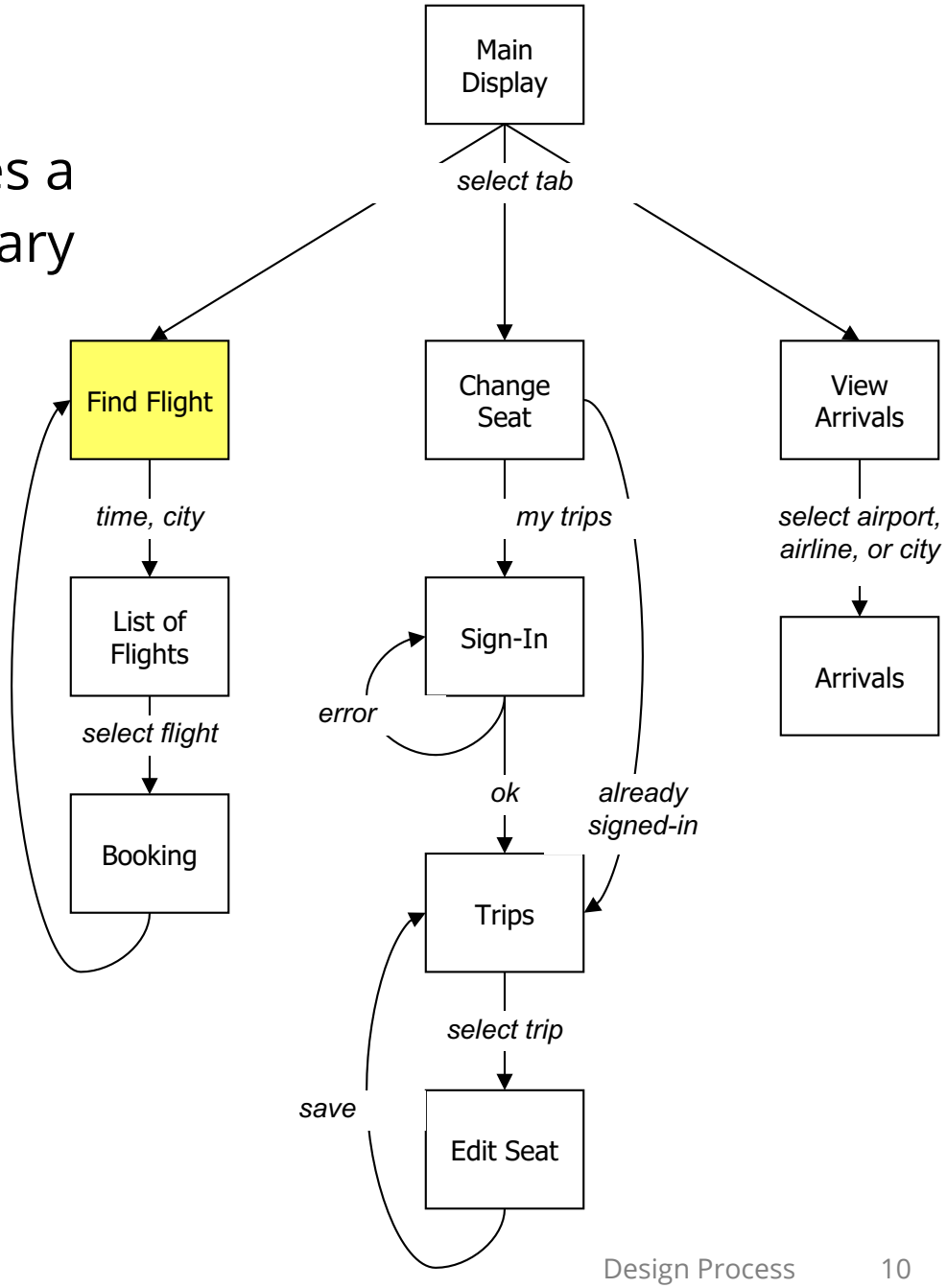
Interaction Sequences

- Illustrate exemplary interaction paths (with alternate paths)
  - like a flow chart, but focusing on interface logic, not Model logic
  - "boxes and arrows"



Interaction Sequence Example

- Labeled boxes and arrows are usually enough, but sometimes a more extensive visual vocabulary may be helpful.



Visual Vocabulary

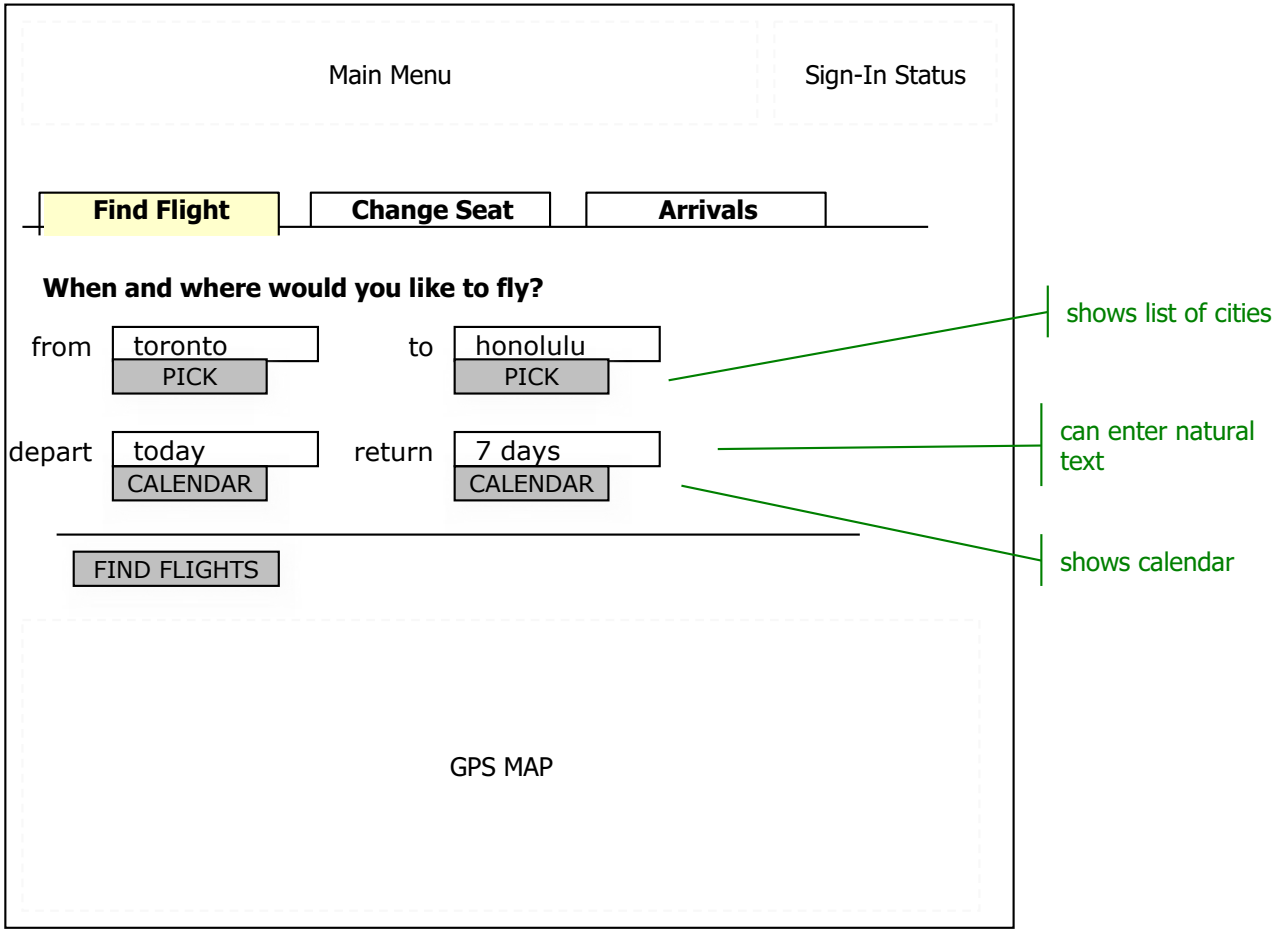
- A standard set of graphical symbols and notation to describe something.
- Qualities of a good UI design visual vocabulary (from JJG)
  - Whiteboard-compatible
  - Tool-independent
  - Small and self-contained
- The UML tends to be too formal, too “high ceremony”
- Jesse James Garret’s visual vocabulary good, but web-centric
  - http://www.jjg.net/ia/visvocab/
- Find a consistent visual vocabulary that works for you.

Interface Schematics (“wireframing”)

- Includes enough detail so that someone could begin designing and implementing system logic
  - visual arrangement of content/information
  - functionality of interface (widgets, forms, data)
  - navigational and content elements ordered to convey structure and meaning
  - indicates the relative significance of all elements
- Interface schematics are not the same as:
  - graphic design mockup
  - paper prototypes

Interface Schematic Example

Find Flight Schematic:

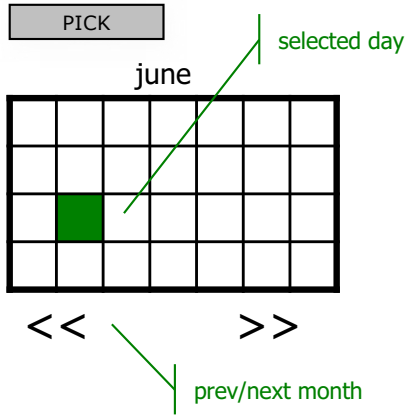


also fine to hand-sketch these, but sometimes using a simple drawing program like PowerPoint is faster if there are revisions.

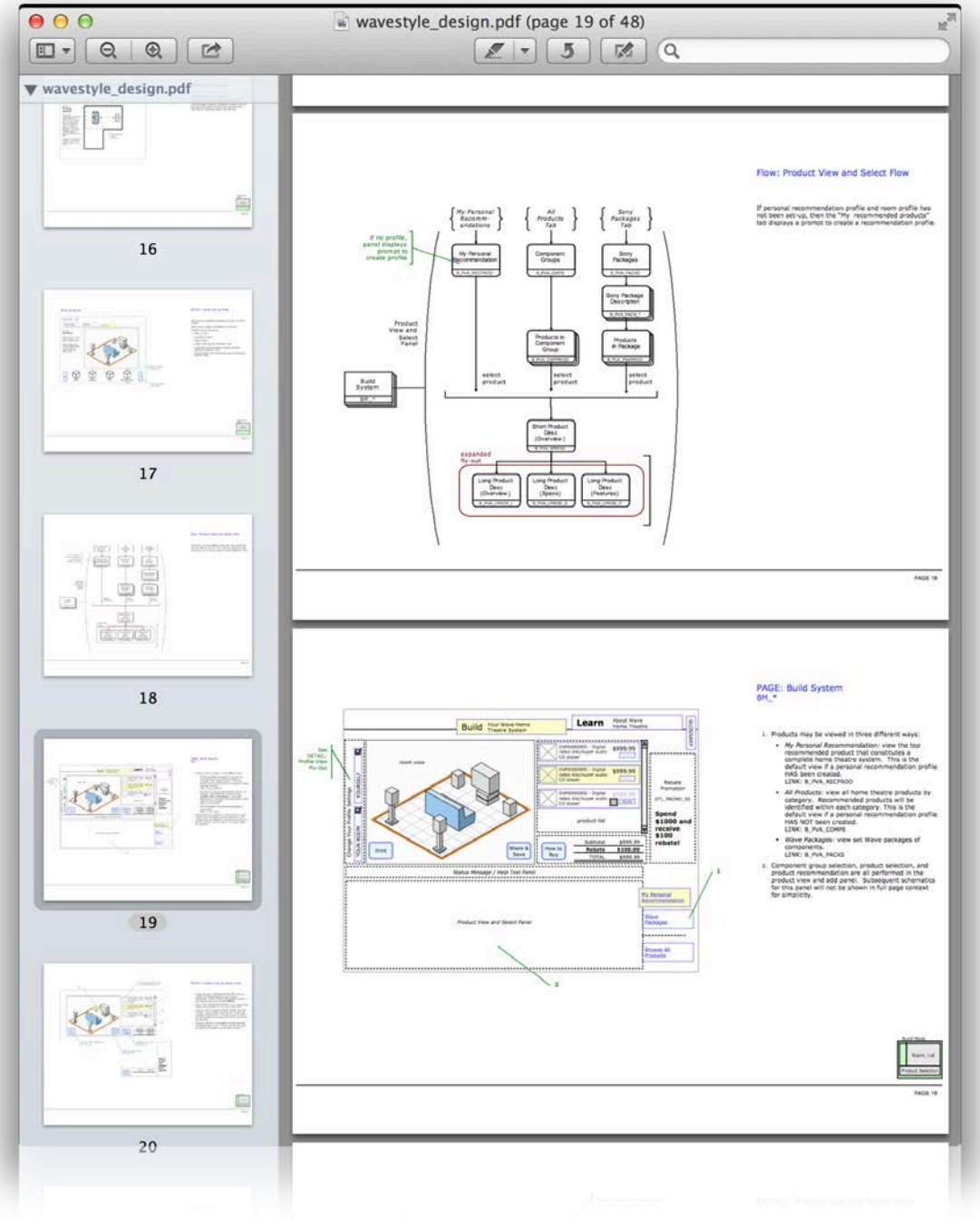
Sign-In Status Widget Schematic:

Note different sign-in and sign-out states.

Pick Widget Schematic:

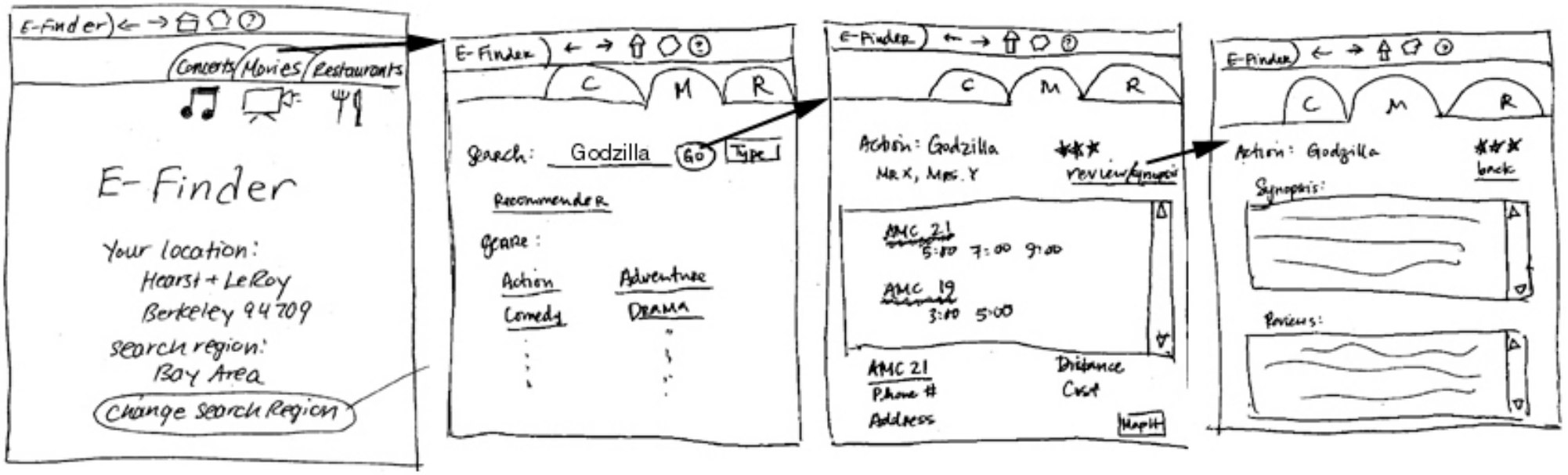


Example Design Document



Storyboards

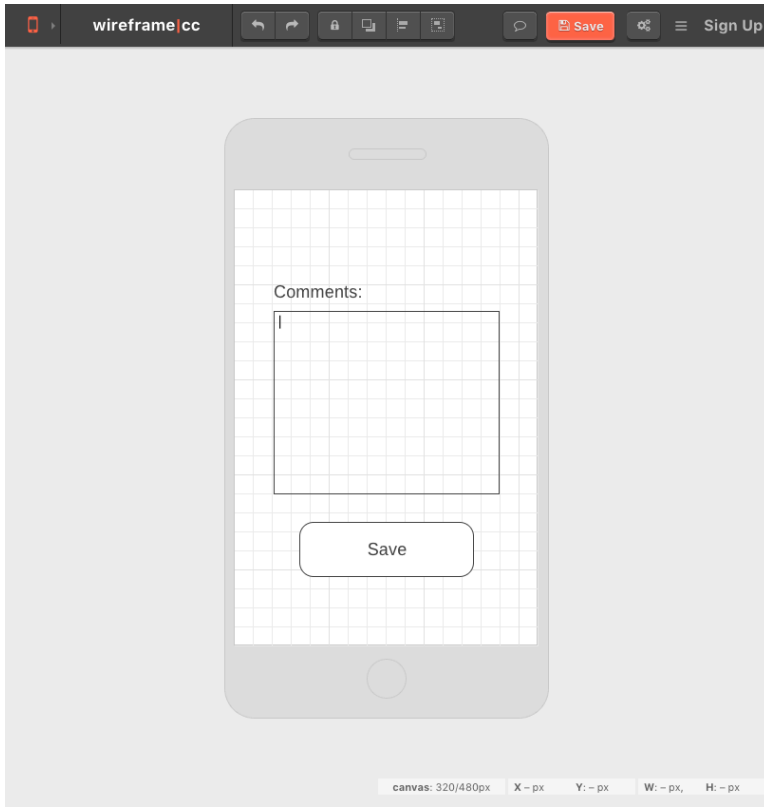
- Quick way to sketch out sequence and schematics
  – could form the basis of a paper prototype



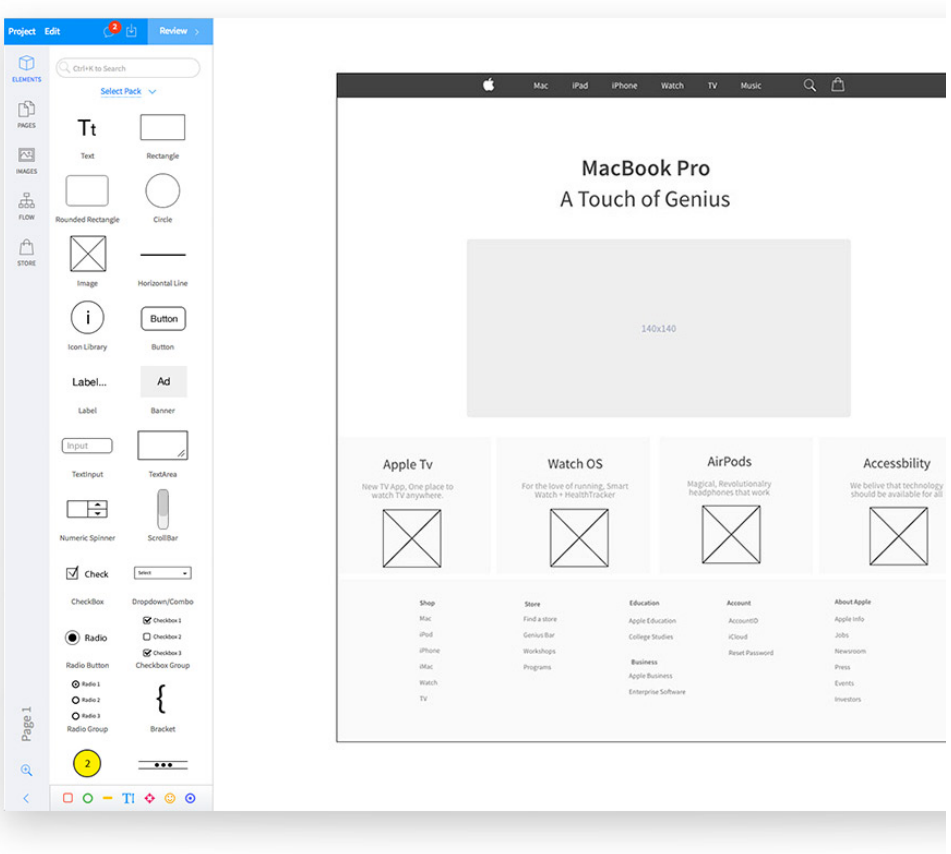
from James Landay's HCI lecture notes

Interaction Design Diagramming Tools

- Many purpose-built drawing tools available



https://wireframe.cc

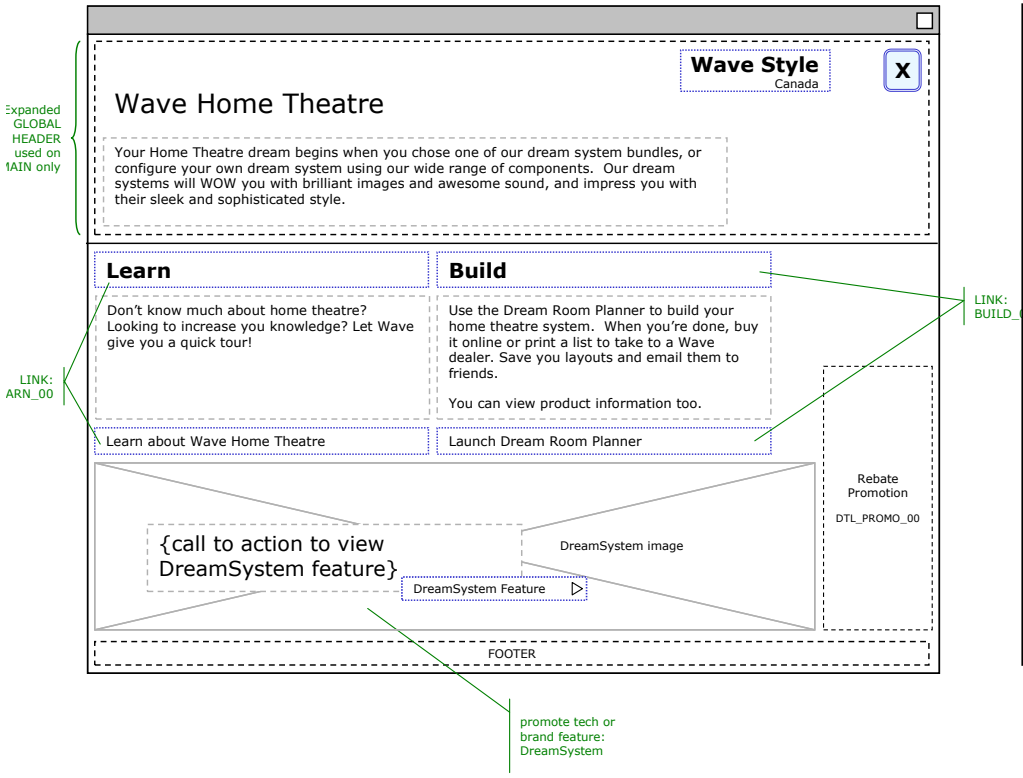


https://www.mockflow.com/

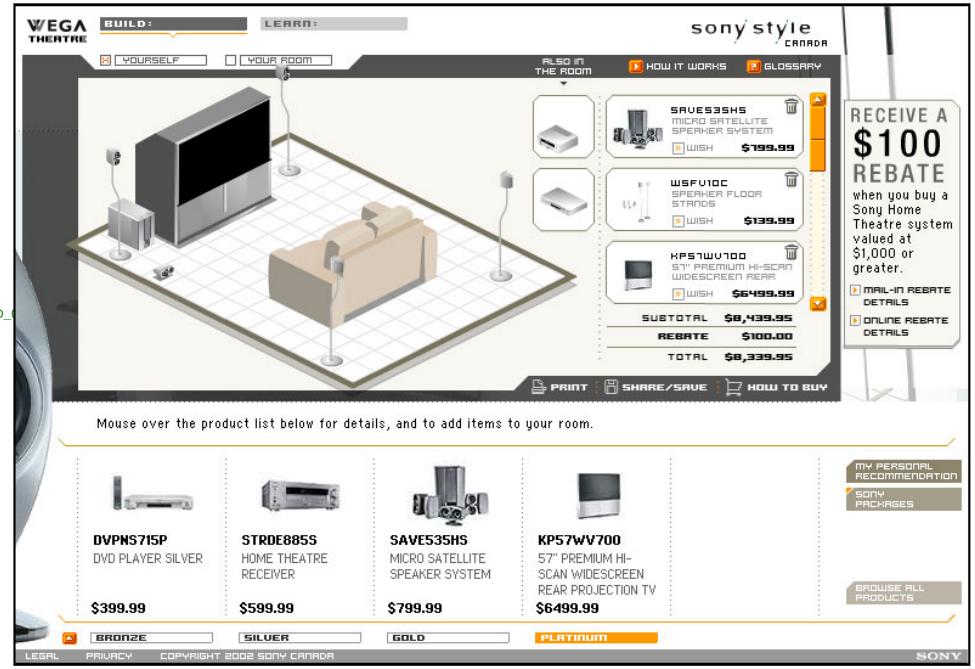
Graphic Design Mockups

- colour, layout, type, shape

schematic



mockup



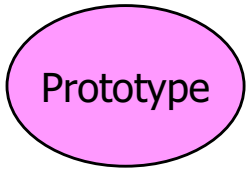
Graphic Design Mockups

- colour, layout, type, shape

schematic

mockup

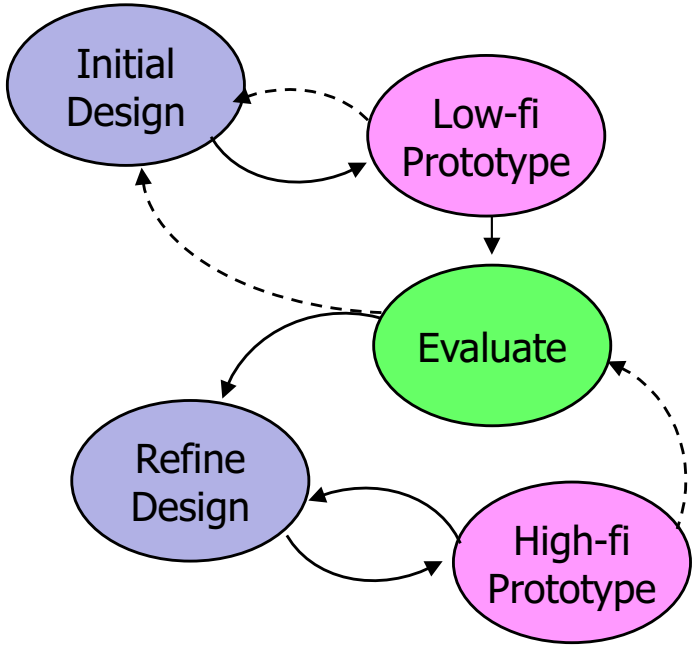
Prototypes



- A limited representation of a design that allows people to interact with it and to explore its suitability
  - Goal: Maximum Feedback for Minimum Effort
- Examples:
  - paper sketches/collage with “human computer processor”
  - slide show or video simulating the use of a system
  - physical models (wood, cardboard, etc.)
  - software or hardware with limited functionality

Prototype Objectives

- Build “working” prototype from design work
  - to aid in discussions with stakeholders
  - to help communication of ideas among team members
  - for user evaluation
  - for technical/layout testing and fine-tuning



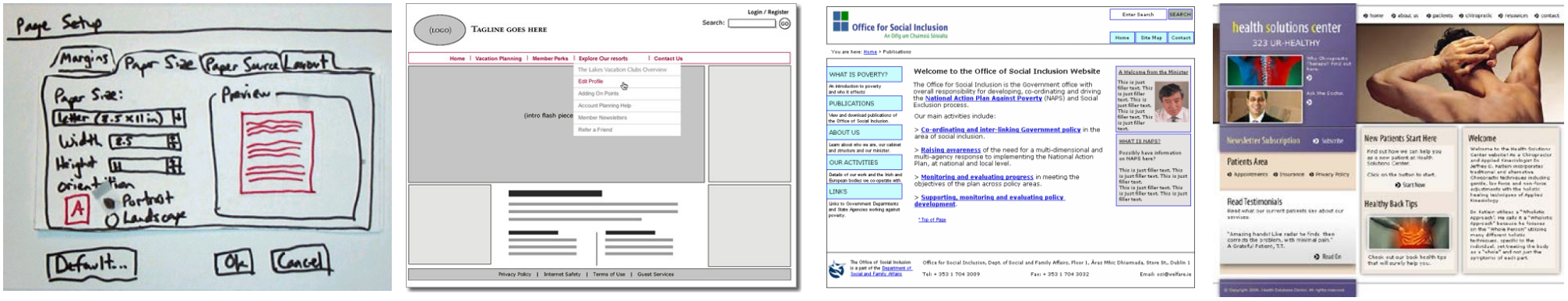
ideally, lots of iteration with design and evaluation

What to Prototype?

- Difficult, controversial, and critical areas
- Possible aspects to test:
  - Concepts and terminology
  - Navigation, task flow
  - Mental models
  - Documentation, help
  - Layout, colours, fonts, graphic elements, “brand”
  - Custom widgets
  - Performance metrics
  - Technical feasibility
  - Usefulness

Prototype Fidelity

- Faithfulness of prototype appearance and performance to final product.
  - Low fidelity: prototype doesn't look much like final product, operation of the prototype may be simulated and slower.
  - High fidelity: prototype looks and operates like the real product.



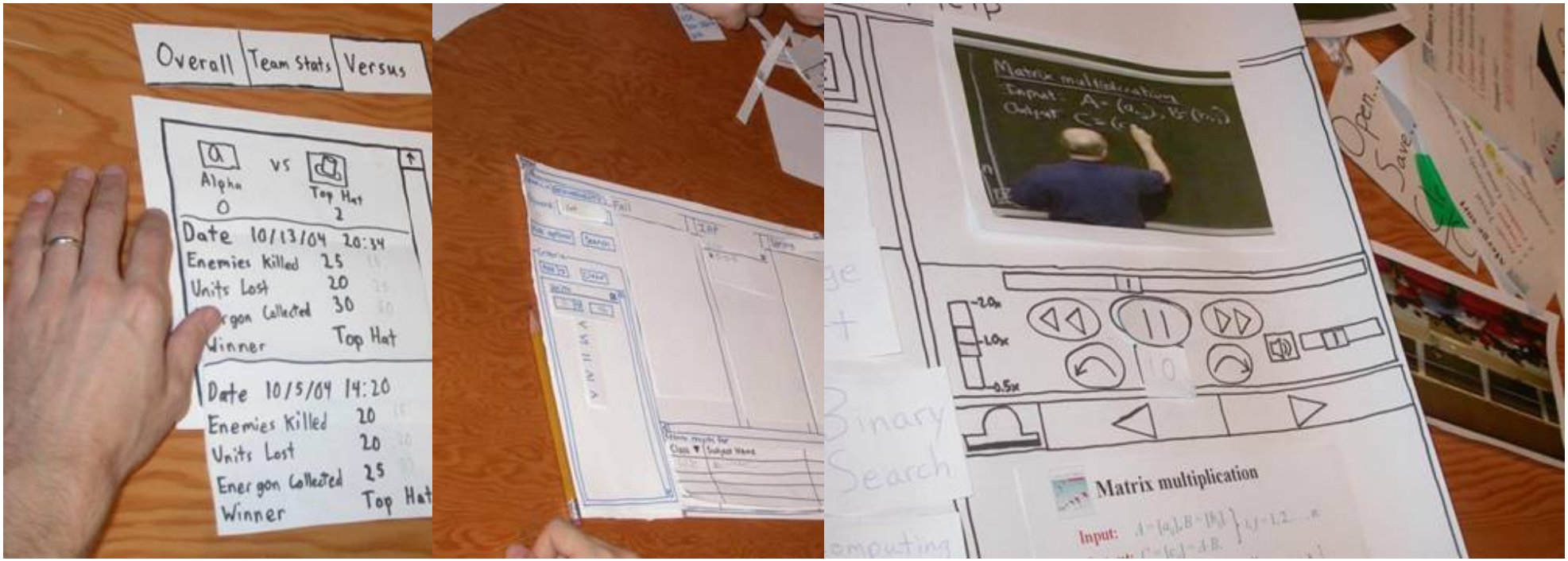
low-fi **fidelity** ***high-fi***

Low vs. High Fidelity Prototypes

- Low-fi prototypes are easy and fast compared to high-fi
  – faster, easier enables more iteration
- Low-fi prototypes are more creative
  – faster, less detail encourages experimentation
- High-fi prototypes can give false sense of completeness
  – people may critique it as a finished product (which it isn't)
- High-fi prototypes are more accurate
  – comments are directly related to final product design

Paper Prototyping

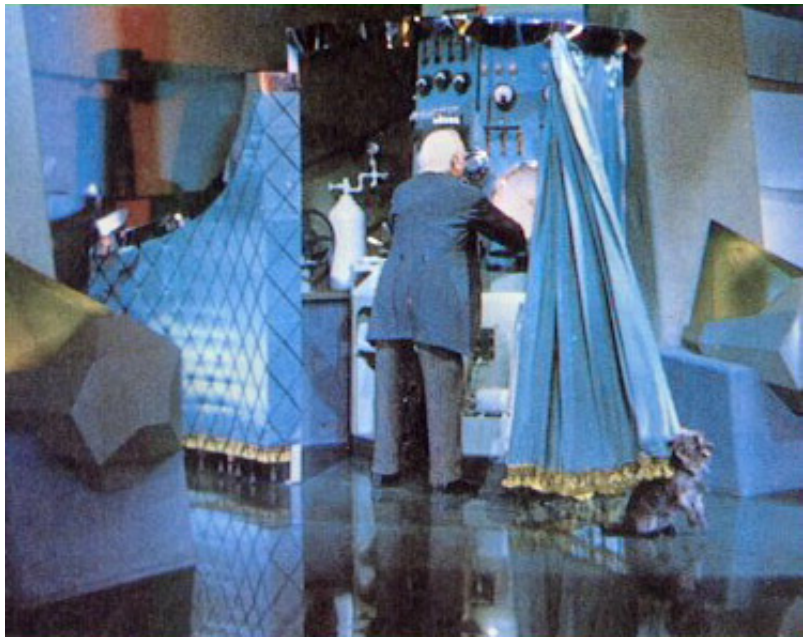
- "Usability Testing where representative users perform realistic tasks by interacting with a paper version of the interface that is manipulated by a person playing computer, who doesn't explain how the interface is intended to work" (Snyder, 2003, p. 4)



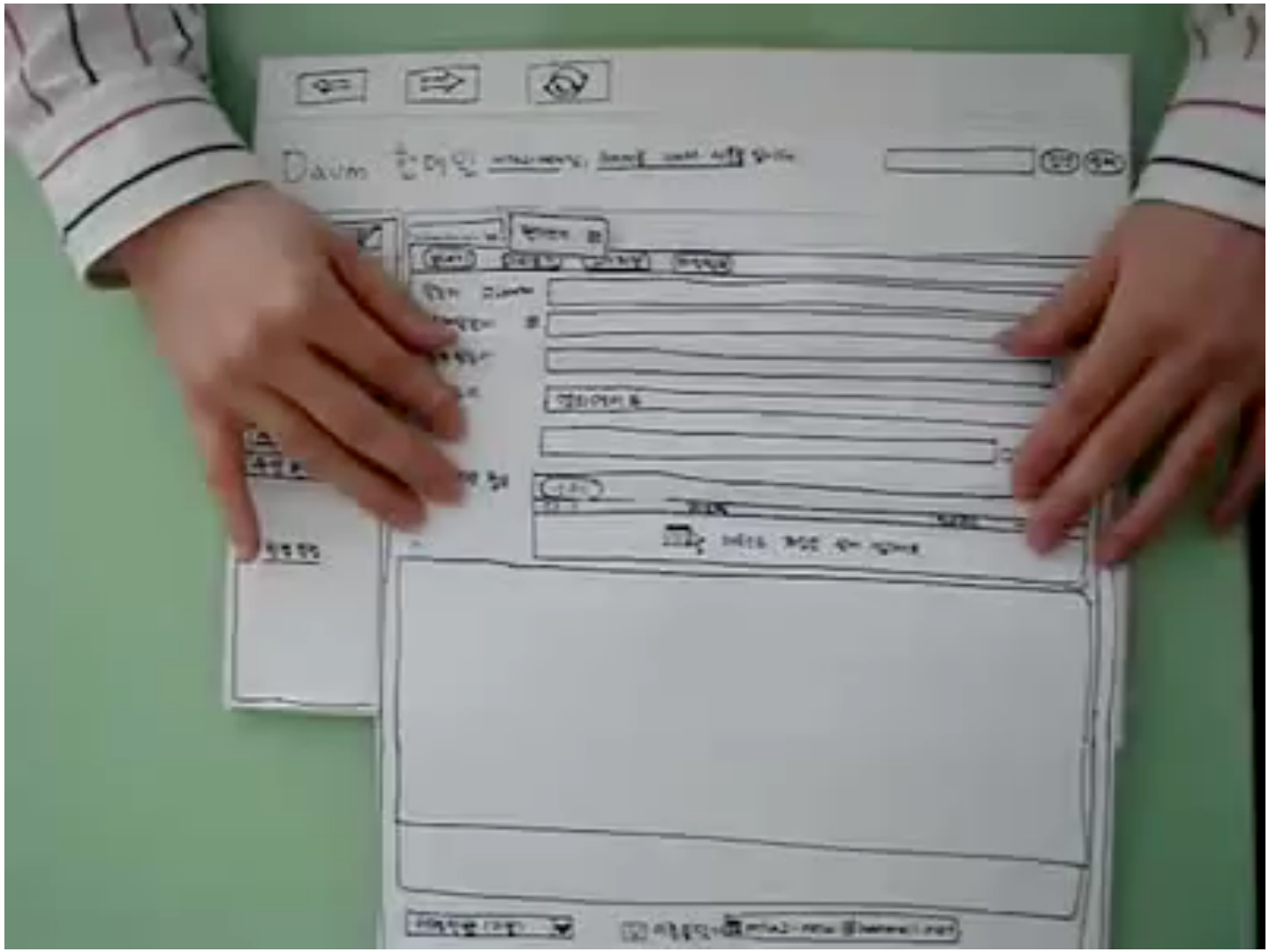
http://groups.csail.mit.edu/graphics/classes/6.831/paper-prototypes/index.html

Wizard-of-Oz Technique (WoZ)

- Evaluate unimplemented technology by using a human to simulate the response of a system
- Much easier to do WoZ simulation than implement complex features like speech recognition



"The man behind the curtain."

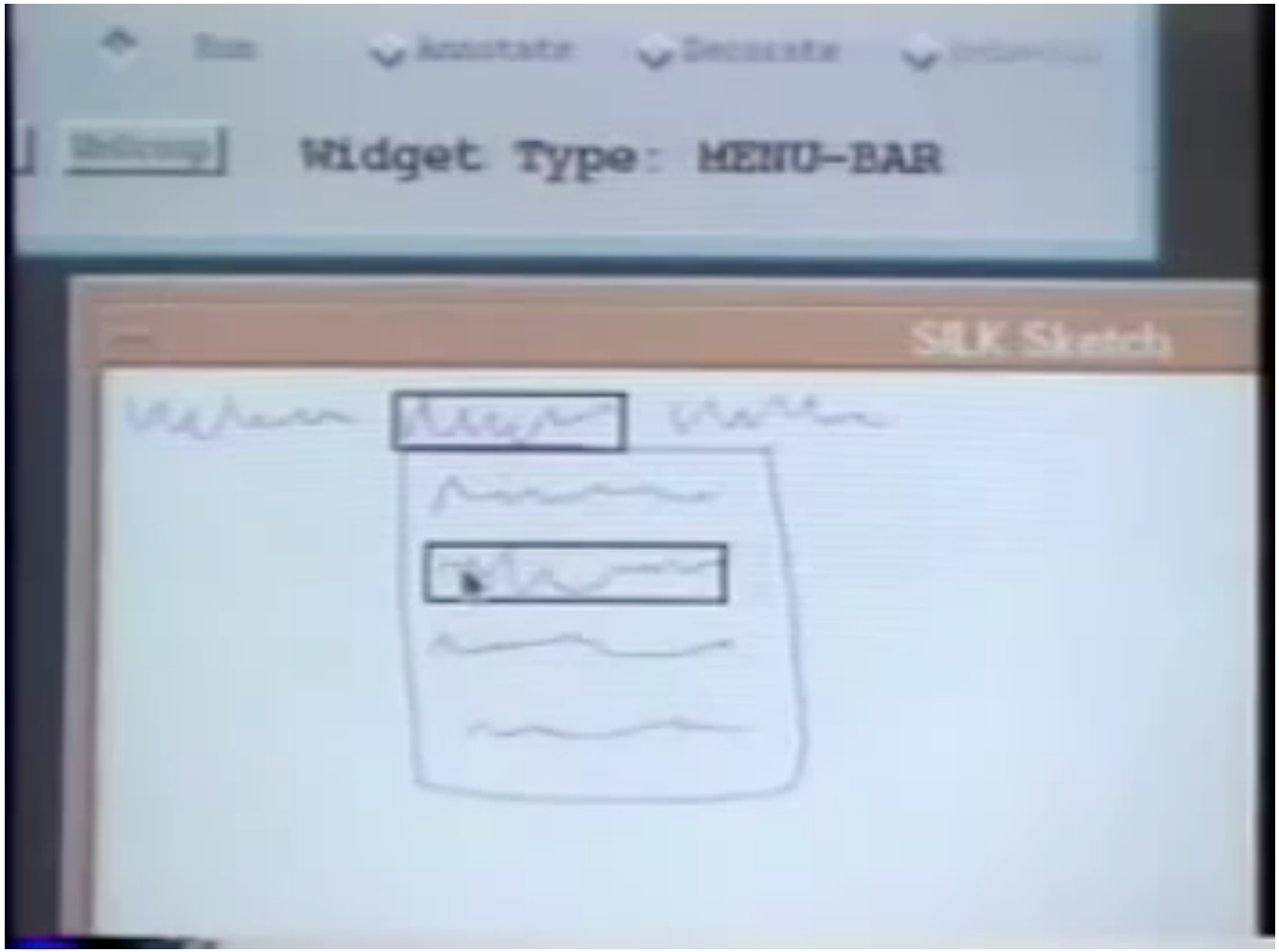


Paper Prototype Evaluation Example

- http://youtu.be/GrV2SZuRPv0

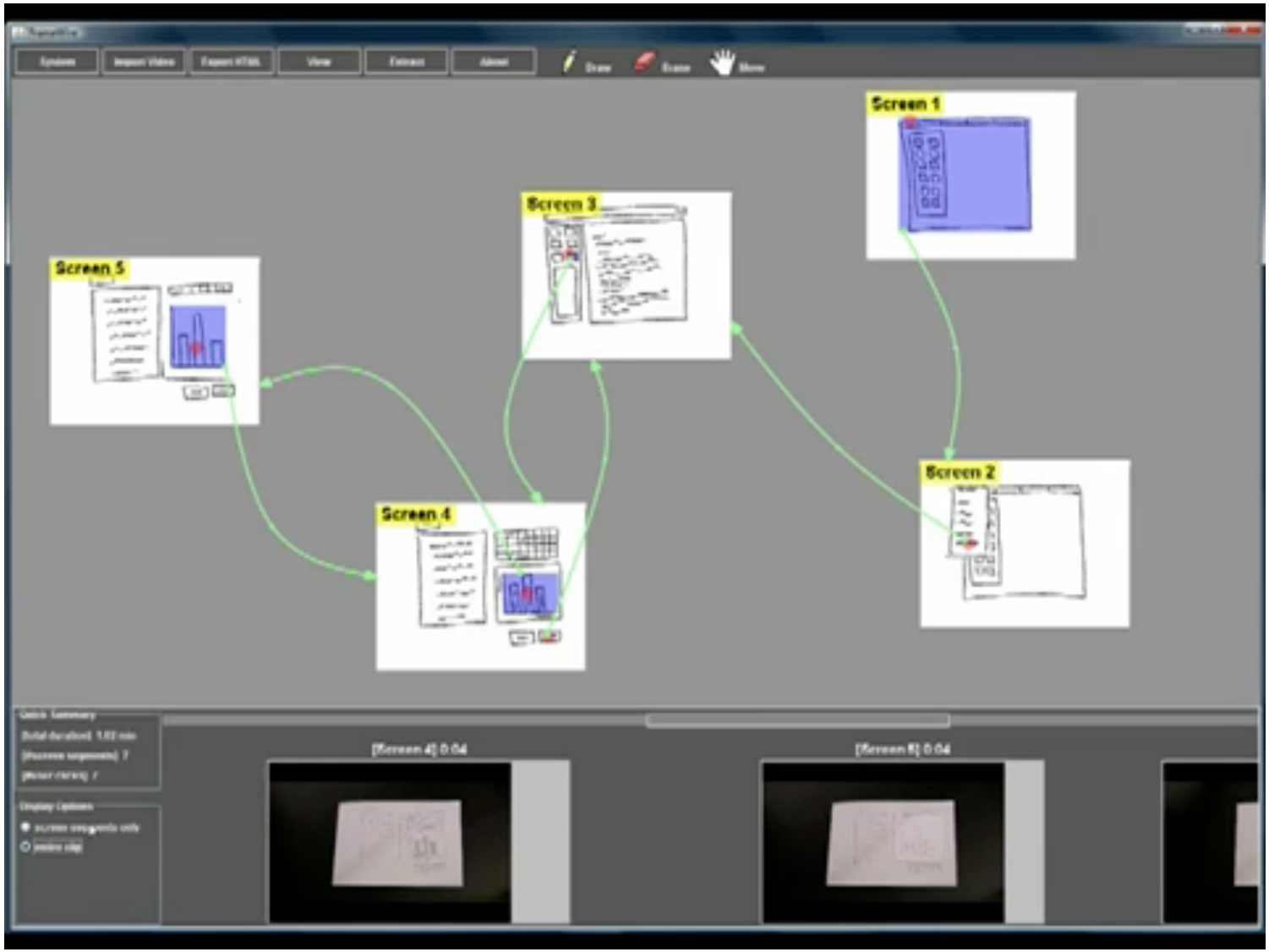
Paper Prototyping Image Quality

- Rendering: How good should it look?
  - Good enough to elicit feedback about critical/risky design issues
- Creation speed is of the essence!
- Modifications - when?
  - While user is present, get immediate feedback. Make changes without loosing users' context. Allow user to make changes?
- Coding effort in paper prototyping is always zero
  - No code to write or rewrite until design is stabilized



Silk: Interactive Sketching, Landay et al. (1996).

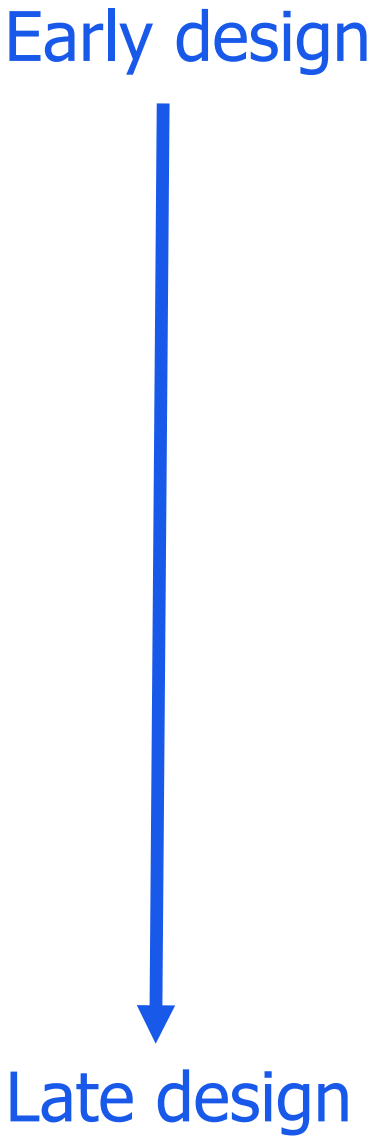
- http://youtu.be/VLQcW6SpJ88



FrameWire, Li et al. (1996).

– http://www.youtube.com/watch?v=q6pGIOhr3d0&feature=youtu.be

When to Prototype



Rough out interface style

Task centered walkthrough and redesign

Low fidelity paper prototypes or physical models

Fine tune interface, screen design

Heuristic evaluation and redesign

Medium fidelity interactive prototypes

Usability testing and redesign

High fidelity interactive prototypes / restricted systems

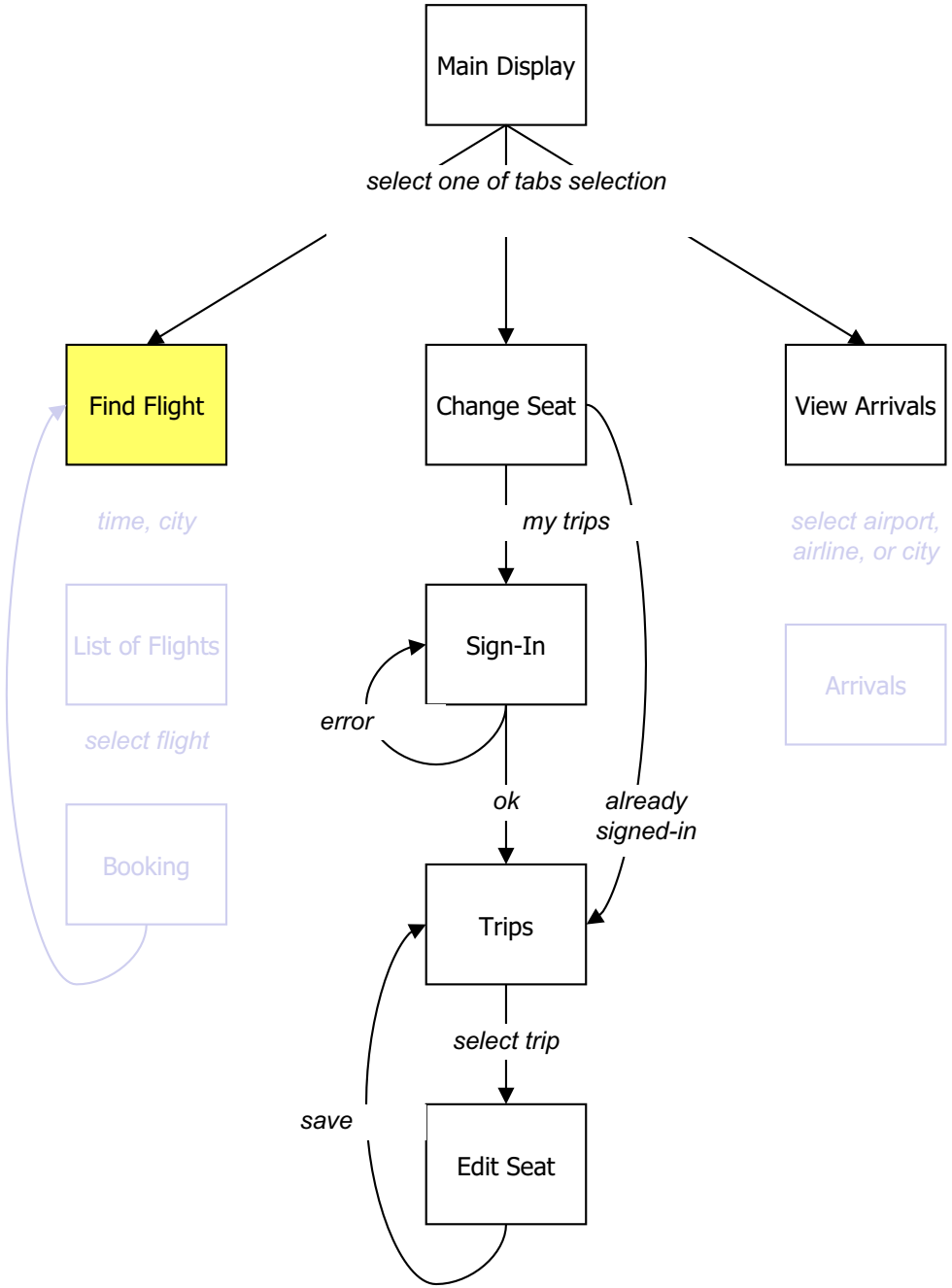
Limited field testing

Working systems

Alpha/Beta tests

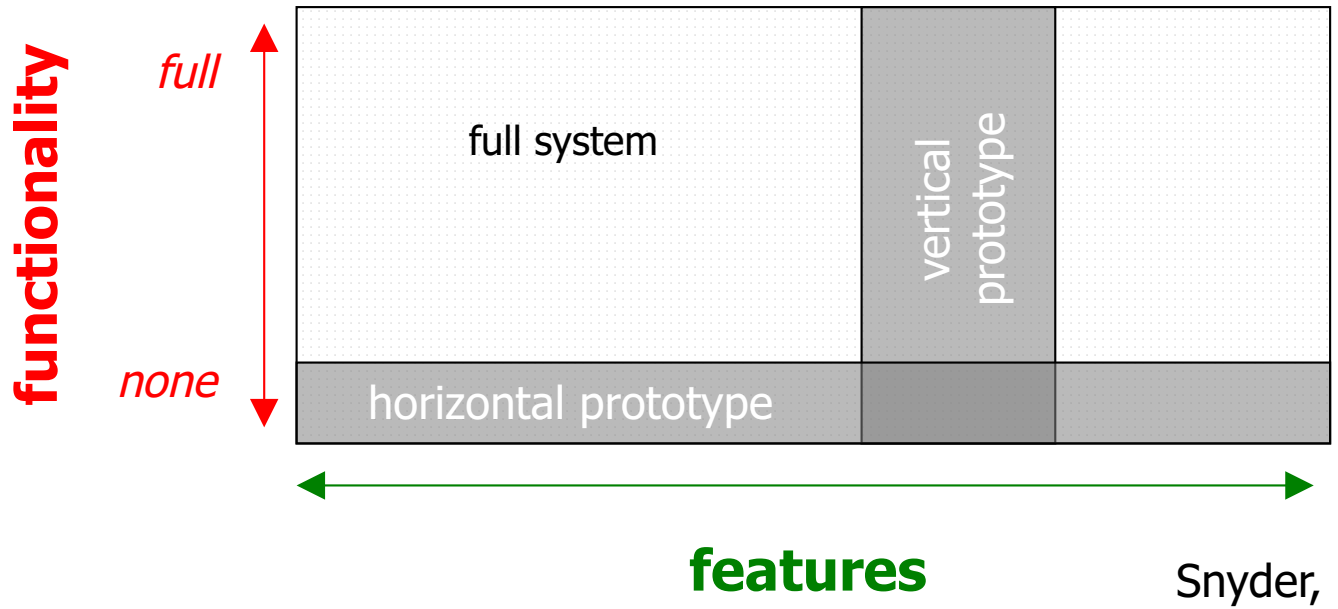
Breadth vs. Depth

- Your prototypes will have some “dead ends” and areas without any functionality
- The scenarios which you want to test will be more complete



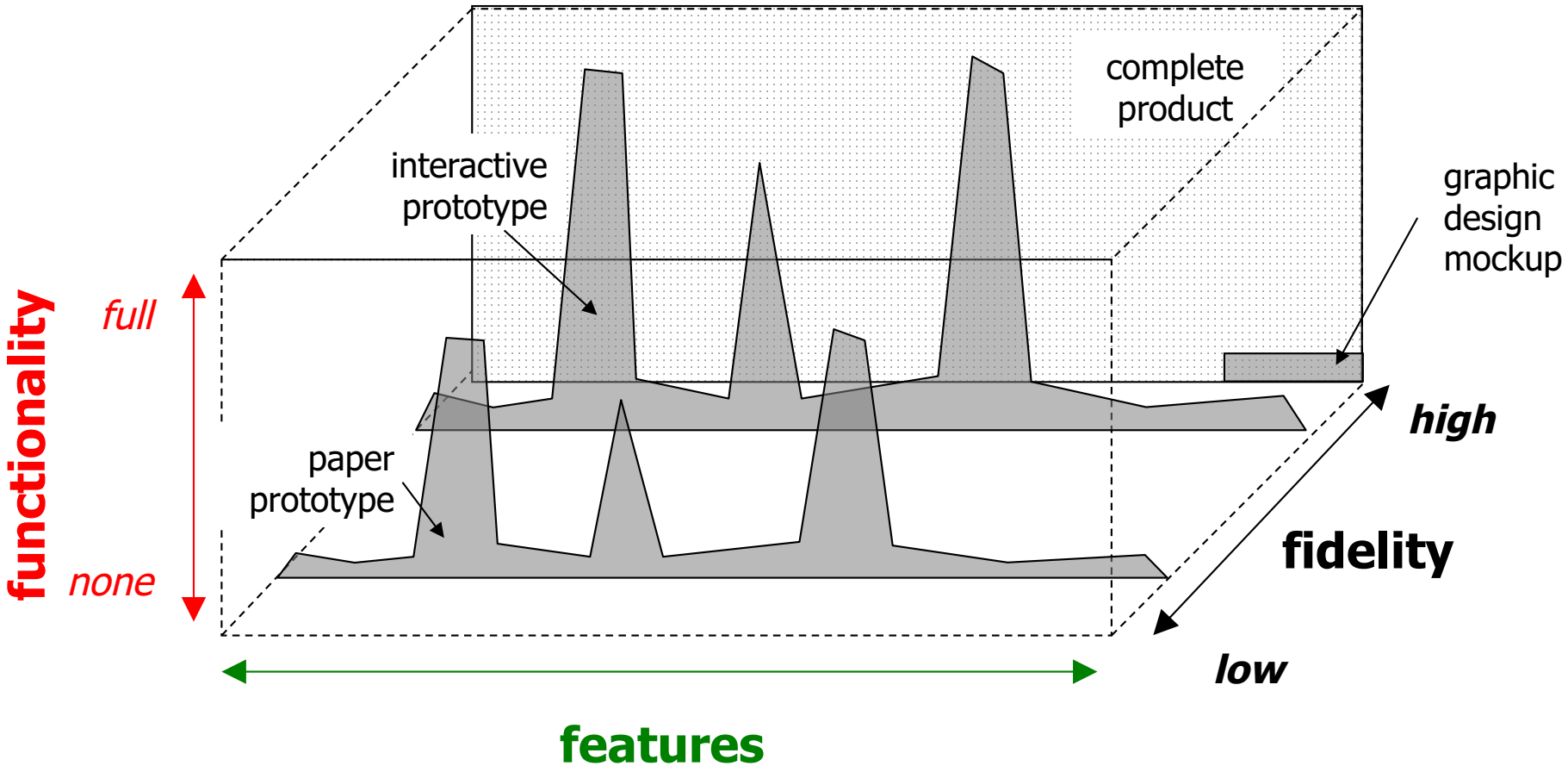
Comprehensiveness

- The amount of features and functionality in the prototype compared to the final product.
  - Functionality (Depth): amount of interactivity and functionality
  - Features (Breadth): amount of features, options, and commands.



Snyder, 2003, pg 260

Prototype Dimensions



Physical Models

- Test ergonomics and form-factor: size, weight, feel (touch)
  – important for specialized devices (small and large)

Balsa wood mock-up

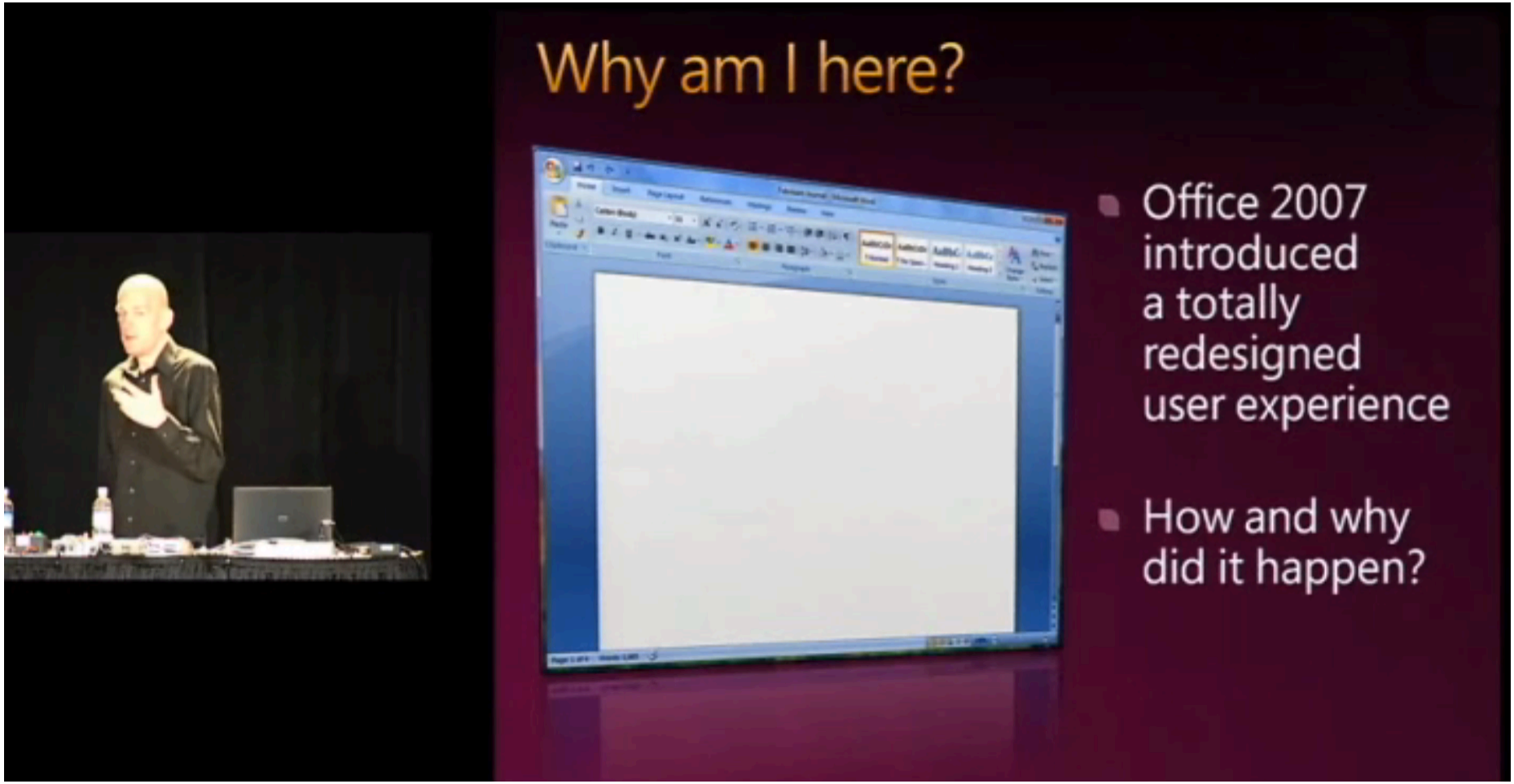
Stereolithography Mock-Up
with Balsa Wood Grip

http://www.canon.com/camera-museum/design/process/camera_design/4.html

Evaluate

Aspect to Test	Paper Prototype	Interactive Prototype	Physical Model	Storyboard/ Scenario	Graphic Mockup
Concepts & terminology	●	●		●	
Navigation, work & task flow	●	●	*	*	
Content	*	●	*		●
Documentation, help	●	●			*
Requirements, functionality	●	●	*	●	*
Screen layout	●	●	●		●
Brand	*	●	●	*	●
Colours, fonts, graphic		●	●		●
Widgets & controls	*	●	●		
Time, performance		●			
Real-world use		●	●	●	
Technical Feasibility		●			

adapted from Snyder, 2003, pg. 268



The Story of the Ribbon (1:30)

- http://channel9.msdn.com/Events/MIX/MIX08/UX09