UI Design Process

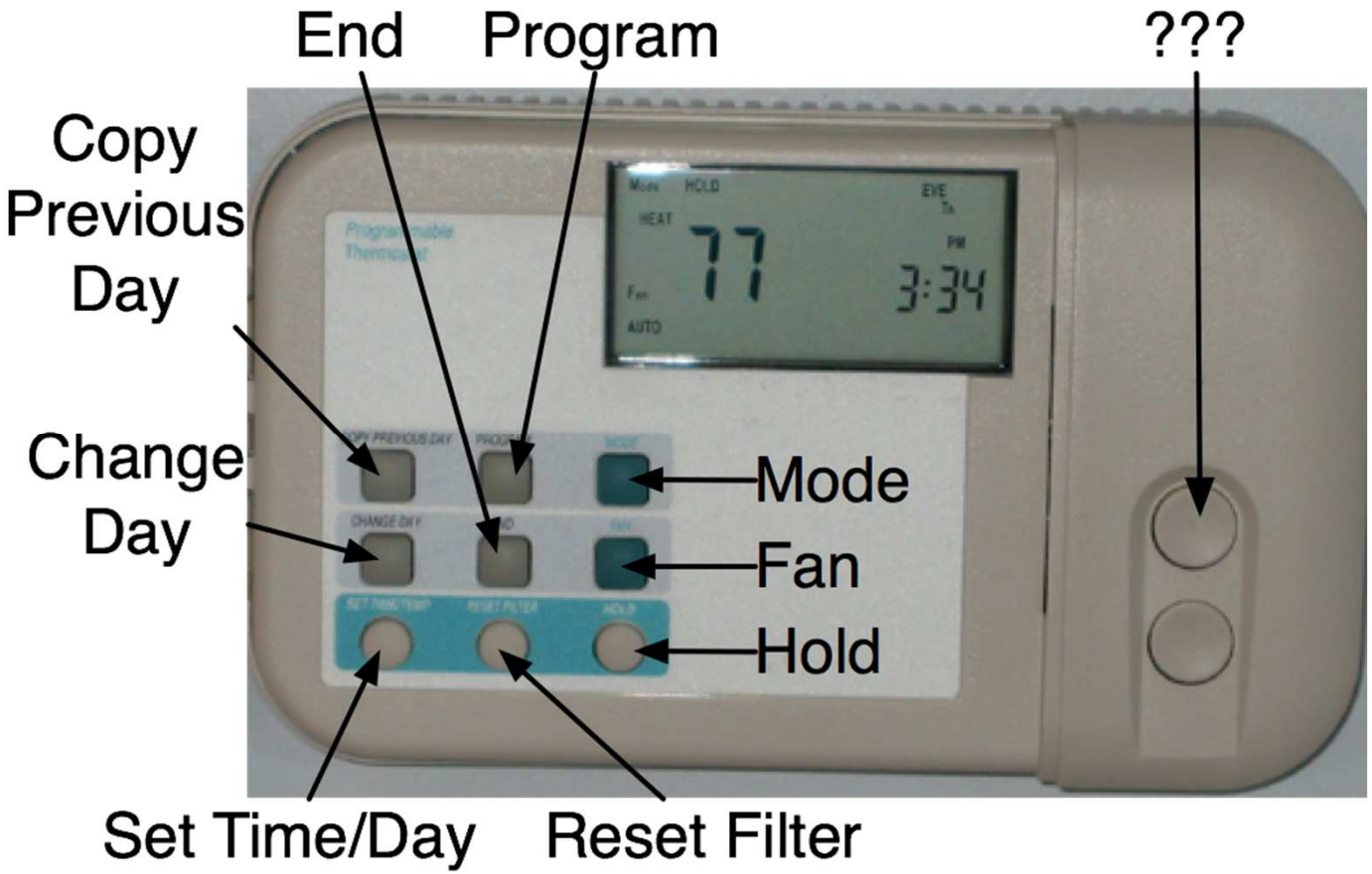
Overview

- User Centered Design: Buzz-words, methodology, state of mind
- Important components in User Centered Design
- Development Process
  - Understanding Users: Scenarios, Functions, Prioritize, Usage Patterns
  - Design the UI: Identify/Design components, Distribute, Test

- In more detail in later lectures

Used Centered Design is a...

- Buzz Word



Used Centered Design is a...

- Methodology

- Developed at...
- Basic flow:
  - User studies
  - Implementation
  - Usability studies
- Problems:
  - Time...
  - Changes...

Used Centered Design is a...

- State of Mind

- *Everyone* involved with the project “wears the head and hands of a user.”

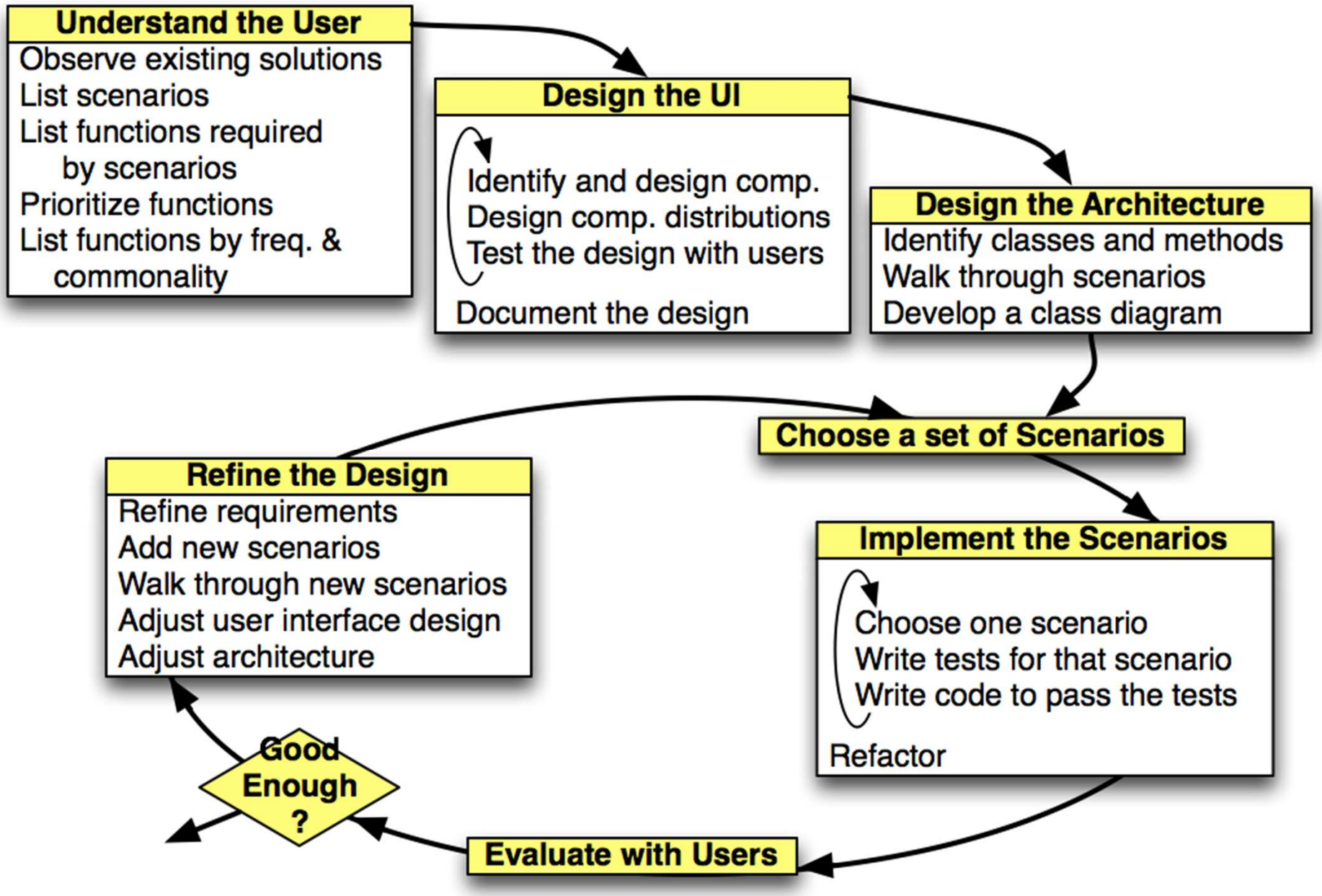
UCD Components (1/2)

- **Understand users' needs**: "Build a product that meets real, observed needs rather than building something because it can be built."
- **Design the UI first**: "Design the UI first, and then design the architecture to support that UI."
- **Iterate**: "The best interaction designer in the world will produce only a decent sketch of a UI design on the first try. A great design requires iteration."

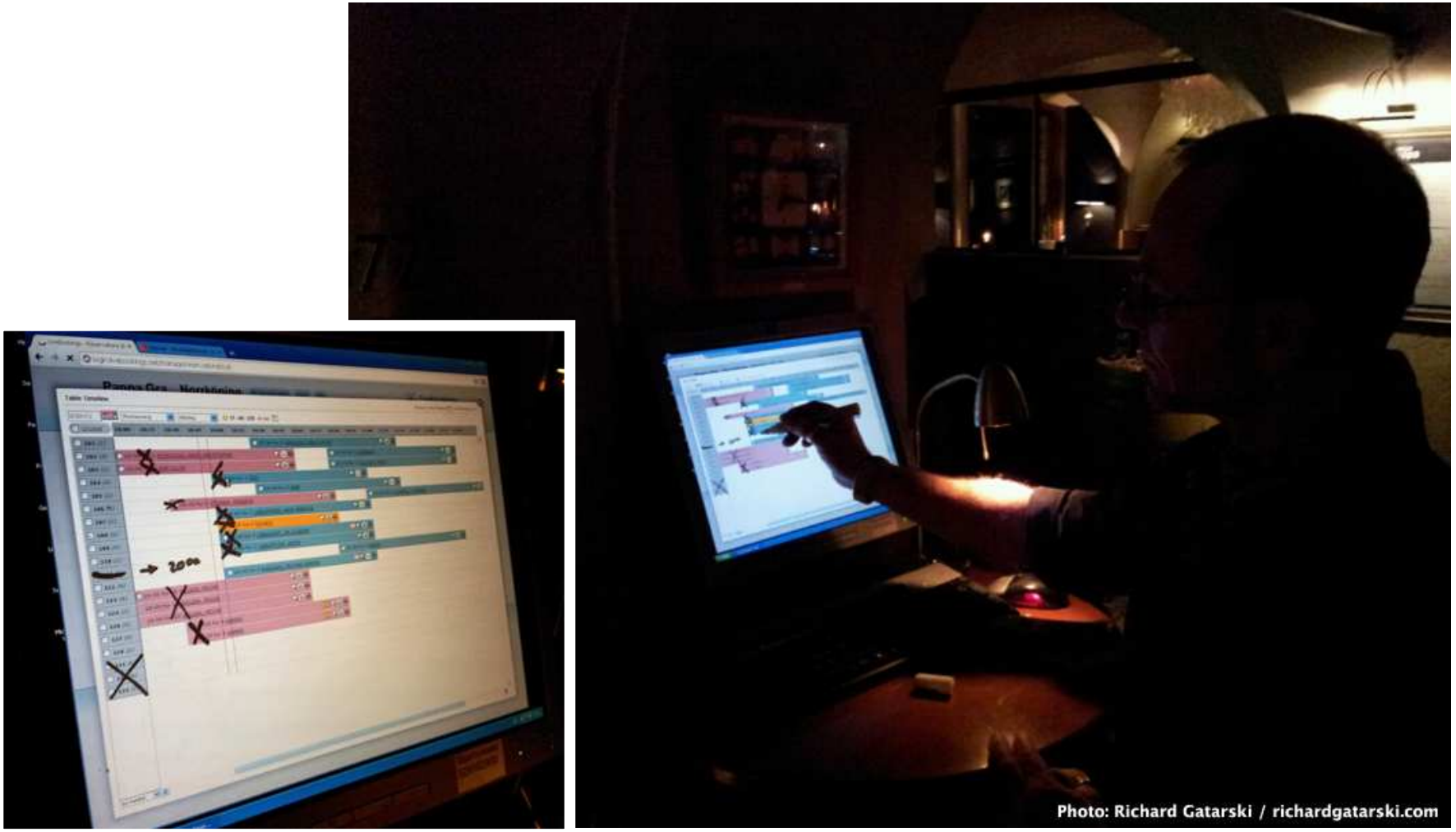
UCD Components (2/2)

- **Use it yourself**: "As you use it, observer all the ways your flow is broken when you do the tasks. You'll find obvious problems…that you can fix while it's still relatively cheap."
- **Observe others using it**: "It is absolutely critical to observe other people using your technology in as realistic a way as possible very early in the development cycle."

  – From *Designing from both sides of the screen*, Isaacs & Walendowski, New Riders Publishing (2002)


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

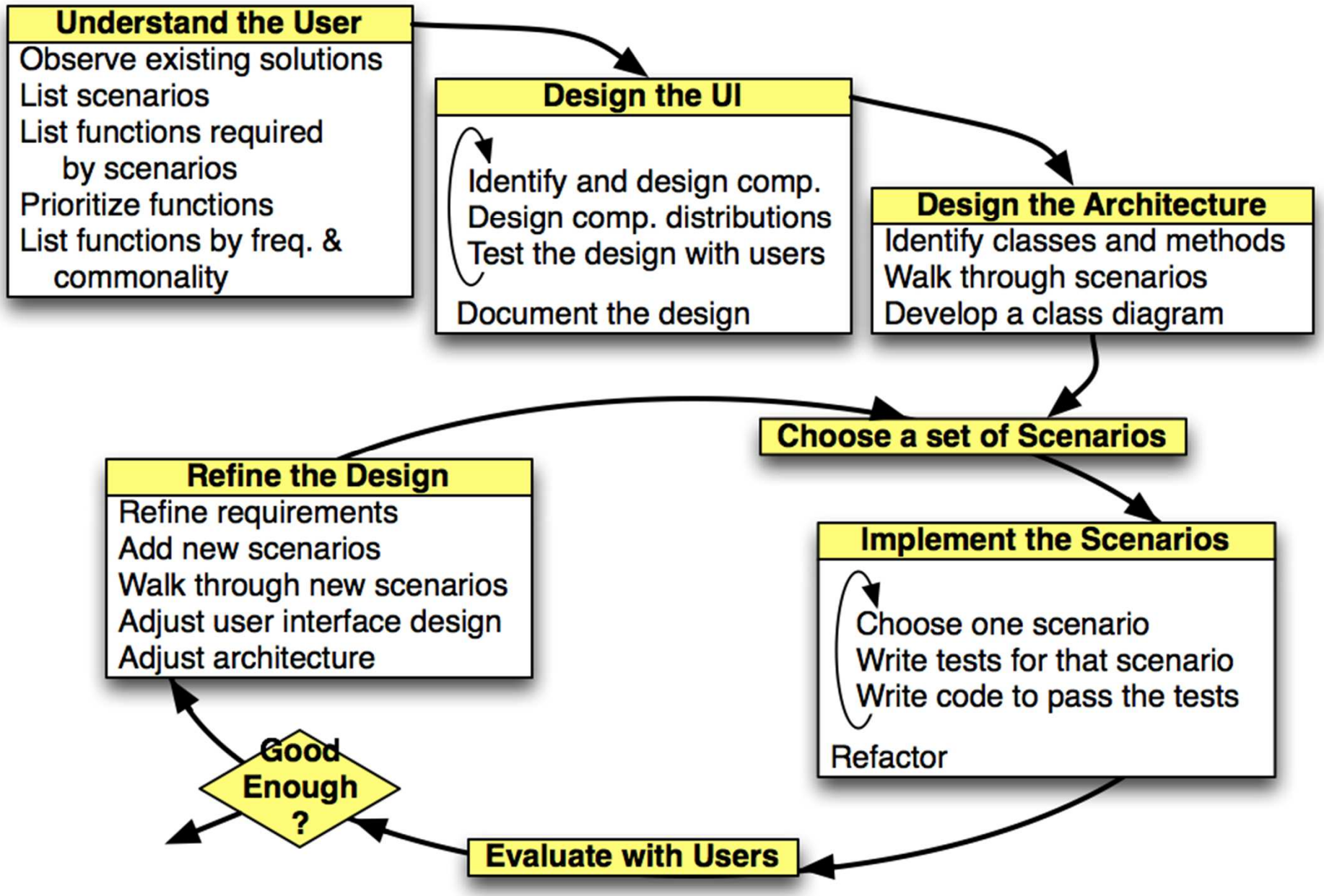
Understand: Observe Existing Solutions



http://javlaskitsystem.se/2012/02/whats-the-waiter-doing-with-the-computer-screen/

Understand: Scenarios

- are stories of people undertaking activities with technology
- are a natural way to think
  - easy to understand (for developers and users)
  - contain sequencing data
- must be refined/elaborated with appropriate detail
  - exactly what user does
  - how UI changes in response
- have pitfalls
  - typical crowds out the exceptional (exceptional uses and users)
  - often fail to catch “oughts”
  - cannot be formalized (also a strength)
- can have variations
- should be retained
  - written vs. memorized vs. generated on demand


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

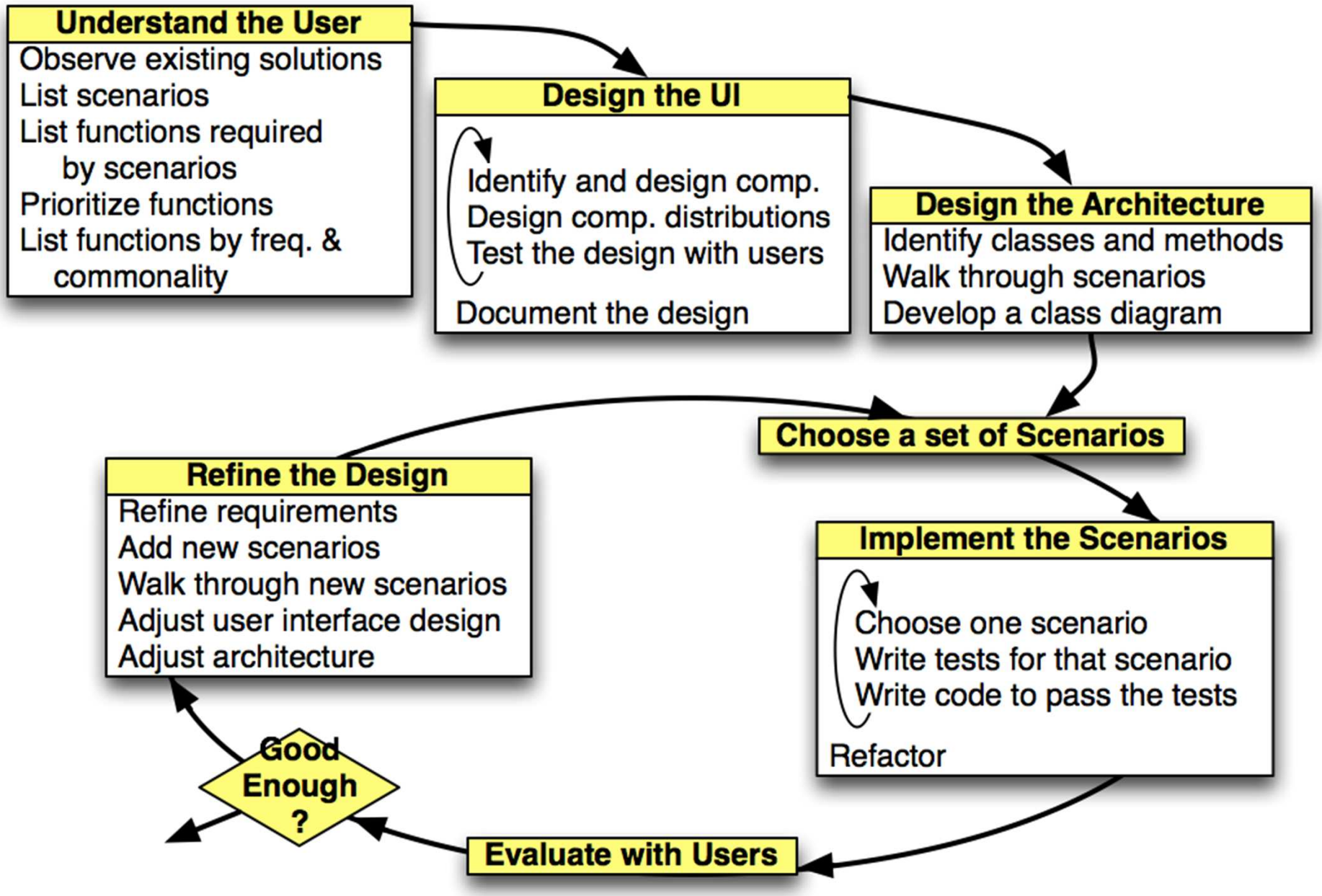
Understand: List Functions

- List functions required by the scenarios
  - Some functions will be required by several scenarios

- Prioritize Functions
  - Core: Needed by early users to do something useful
  - Important: Required before shipping the product
  - Nice to Have:

Understand: Usage Patterns

	By Many	By Few
Frequent	Most people will do this task frequently (nearly every time they use the application). Visible, few clicks	Only some people will do this task, but they will do it frequently. Suggested, few clicks
Occasional	Most people will do this task, but only occasionally. Suggested, more clicks	Only some people will do this task and only occasionally. Hidden, more clicks


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

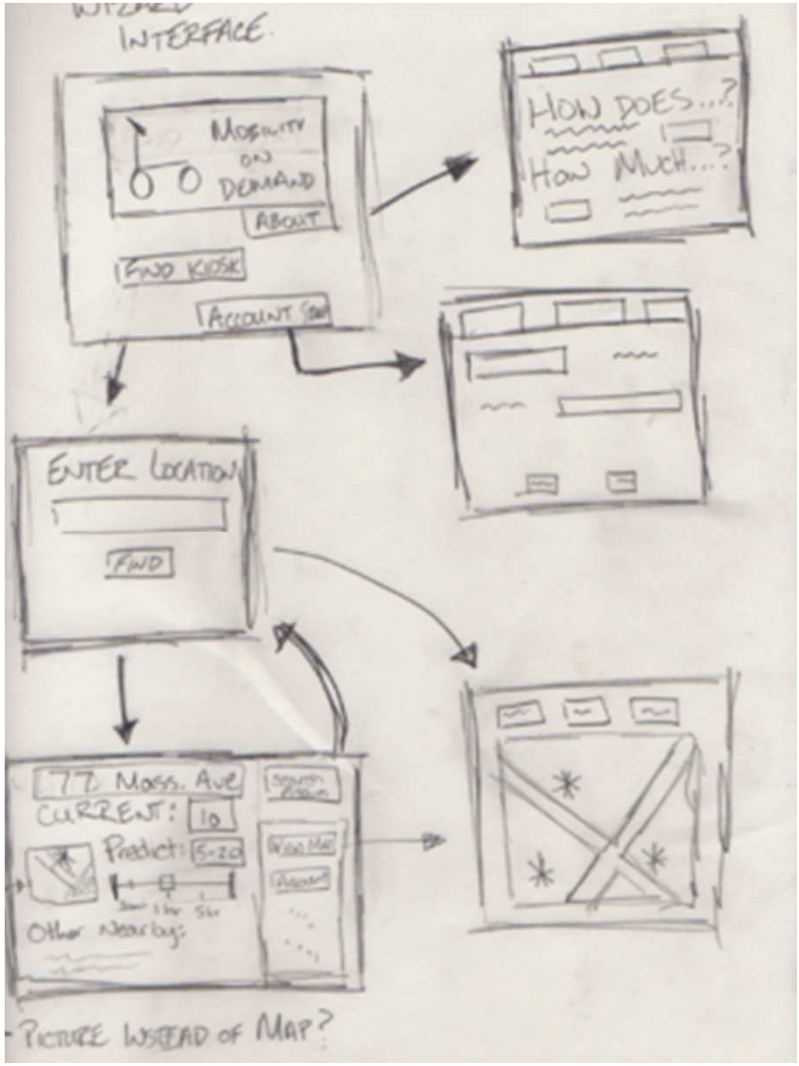
Design the UI

- Identify/design needed components and component types
  - eg: textfields, buttons, etc.
  - Balance functionality with well-understood component types
  - Assign attributes
    - State: what data does it need?
    - Affordances: what can you do with it?
    - Presentation: how does it appear in the UI?
- Design component distribution: temporal and spacial
  - Screen flow
  - Screen layout
  - *Many guidelines -- next lecture!*

- Test the design with users
  - *So important it gets its own lecture!*

Design: Component Distribution

- Temporal distribution:
  - When components appear
  - Flow from one screen to another
- Spatial distribution:
  - Where components appear on an individual screen



CS 449 in brief: Contextual Design

- Explicit process that supports design of software
  - Do contextual inquiry
  - Develop models of work for people you study
  - Consolidate these models to produce a single picture of your user
  - Redesign how user will work with your system as a component
  - Define the overall structure of your system to work with user's new work process
  - Mock-up and test with customers
  - Implement

9 weeks
No computers

Design: Test with Users

- Another lecture!