UCI University of California, Irvine

Informatics 134

Software User Interfaces
Spring 2021

Mark S. Baldwin
baldwinm@ics.uci.edu

4/13/2021

Agenda

1. Designing User Interfaces

2. T2: Scenario Story

3. Next Class

Designing User Interfaces

Thinking About Design

> “Design is a plan for arranging elements in such a way as best to accomplish a particular purpose.”
>
> ——Charles Eames

When applied to HCI

a plan: processes and methods
arranging elements: a naturally creative endeavor
accomplish: through tools or other things
particular purpose: human use and other people-centered concerns

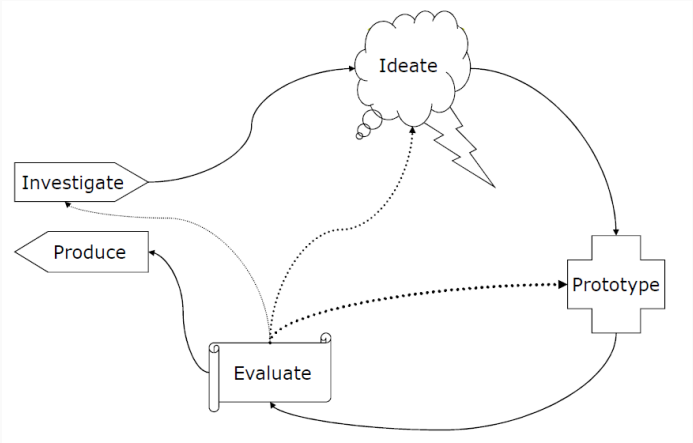
In HCI we talk a lot about users and experiences...

User Centered Design

User **Experience** Design

What do you think they have in common?

Look familiar?



At the heart of any creative endeavor in HCI is the iterative process

We Investigate

We Ideate

We Prototype

We Evaluate

And We Produce

Investigate

- Learn about stakeholders
- Discover goals and needs
- How is it done now?
- What is wanted?
- What else has been tried?

Ideate

Generate **lots** of ideas

Grasp issues and potential solutions

Why?

Ideate

Generate **lots** of ideas

Grasp issues and potential solutions

Why?

- The more ideas the better chance for success
- Systematic evaluation of a large volume of ideas
- Be sure to avoid picking your first idea!

Prototype

- Produce something tangible
- Identify challenges
- Uncover subtleties

Why?

Prototype

Produce something tangible

Identify challenges

Uncover subtleties

Why?

- It's hard to evaluate a thing that does not exist
- Helps your audience understand abstract concepts
- Helps **you** identify future constraints and bring potential obstacles into view

Evaluate

- Discover problems
- Assess progress
- Determine next steps

Why?

Evaluate

Discover problems

Assess progress

Determine next steps

Why?

- Feedback on your direction and ideas
- Increase chance to discover major issues
- Help to resolve disagreements

What are some ways that you can ideate, prototype, and evaluate?

What are some ways that you can ideate?

- ”Idea Oscillation” - gradually iterate through micro-changes
- Immersion
- Generation (keep a design notebook)
- Sketching and storyboarding
- Talk (and listen) to anyone ... everyone
- Modify what already exists
- Mix and match across domains
- Read/watch science fiction!

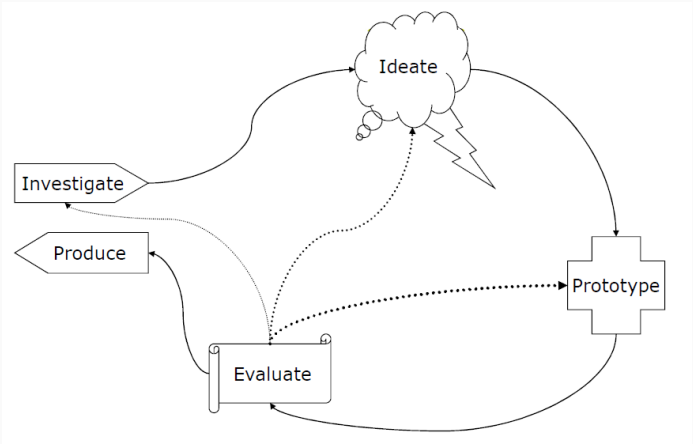
What are some ways that you can prototype?

- Paper prototypes
- Power point
- Software tools

What are some ways that you can evaluate?

- Ask a friend, ask a stranger
- Share with your team
- A/B testing
- Publish low-fidelity online, ask for feedback
- Controlled via lab setting

What's the point?



Designing User Interfaces

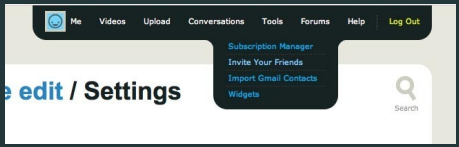
Designing user interfaces is hard!

Designing User Interfaces

UI's are hard to implement...

From a design perspective

From a programming perspective

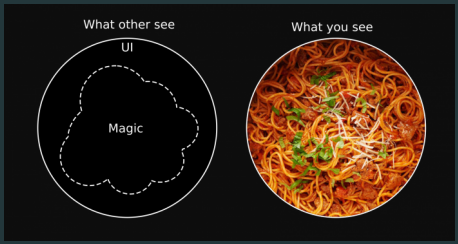



```
var target = document.querySelector('.box');
var player = target.animate([
  {transform: 'translate(0)'},
  {transform: 'translate(100px, 100px)'}
], 500);
player.addEventListener('finish', function() {
  target.style.transform = 'translate(100px, 100px)';
});
```

Designing User Interfaces

From a programming perspective

- Reactive, must respond to difficult to predict human behavior
- Event-based, difficult to model **and** modularize
- Dependent on multi-processing (peripherals, displays, local/remote communication)



From a programming perspective

Must be robust enough to handle:

- Device input
- Video and audio
- Background processes

From a programming perspective

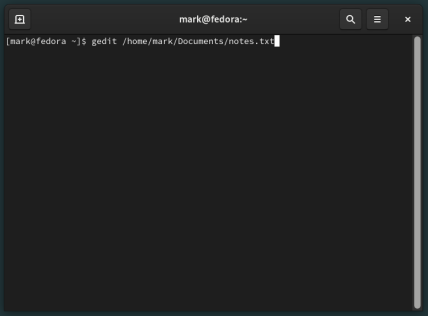
Must be robust enough to:

- Avoid crashes
- Support recovery (help, rollback/undo, escape/abort)

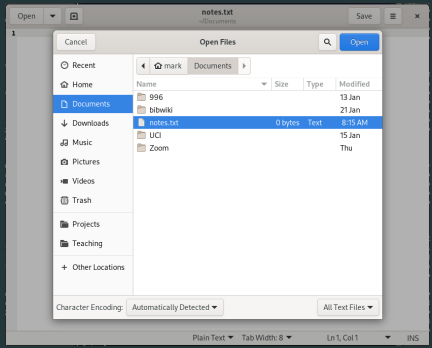
What is going on here?

From a programming perspective

Consider the difference between:



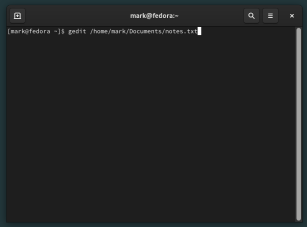
and:

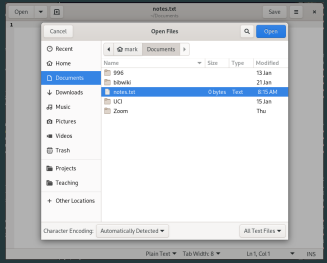


From a programming perspective

Both perform the same action, but the graphical UI must also:

- Support modal
- Cancel (abort/escape)
- Gather and display system resources
- Search
- and many more...





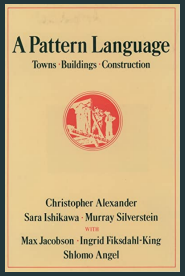
From a programming perspective

Design patterns, to the rescue?

> Design patterns provide a common language upon which designers and developers can reason about intent and function.

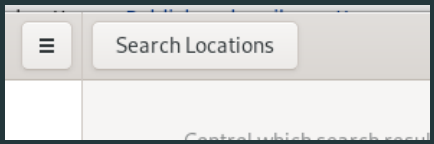
On design patterns

> “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”
>
> ——[Alexander, 1977]

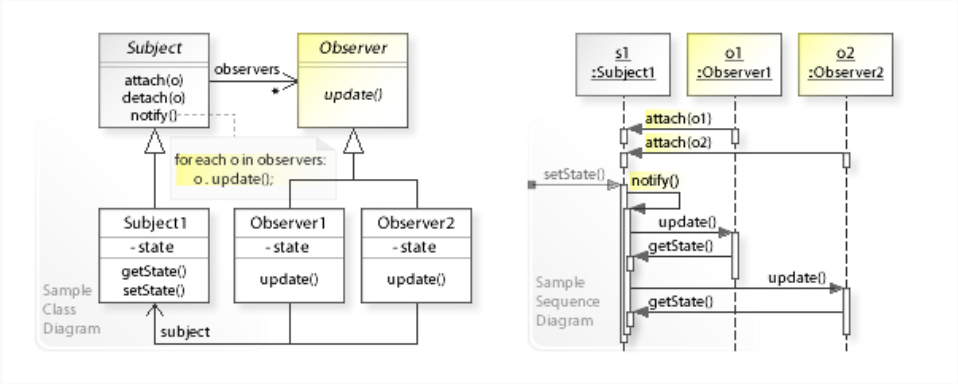


From a programming perspective

UI's manage complexity through design patterns



The Observer Pattern



[Wikipedia, 2021]

From a programming perspective

The Observer Pattern

A standard model for handling event propogation across nearly all UI toolkits

Some examples:

- Microsoft .NET
- TypeScript
- React
- Java

From a programming perspective

But even a simple button is filled with complexity

Examples?

From a programming perspective

But even a simple button is filled with complexity

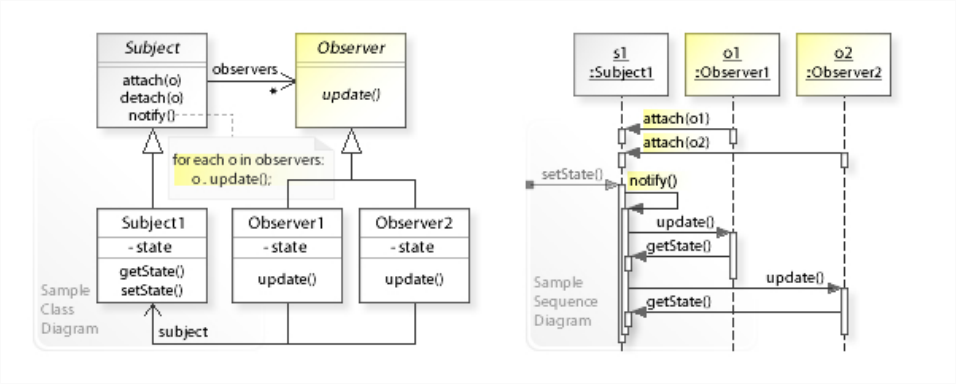
- Idle state
- Hover state
- Mouse up
- Mouse down
- Pressed
- Released
- Hover up/down?
- Idle down?

Design patterns are not perfect

As UI complexity grows, design patterns can lead to code that is hard to learn. The observer pattern, for example:

- Promotes side-effects: Since a subject is decoupled from its observer, an event (click, hover) can have *n* observers...
- Difficult to trace control flow and debug

Observer1, Observer2, ObserverN



[Wikipedia, 2021]

Deprecating the Observer Pattern

Work by Martin Odersky (Scala, Generic Java, many other contributions)

Via Scala.React system, paradigm shift from observer-based to data-flow based model

[Maier et al., 2010]

What can we learn?

Widgets and interfaces require patterns and architectures

Design patterns and architectures can help us communicate and envision how to bring disparate elements together.

How will we learn?

Next week we will start learning how to incorporate patterns like observer into the development of our own widgets

You will collectively apply these skills and others as a team to build a compelling and novel user interface

T2: Scenario Story

Next Class

Thursday

- T1 Presentations
- Keep working on A2 (DUE 4/19)
- Get started on T2 (DUE 4/20)

References

References i

- Alexander, C. (1977).
  A pattern language: towns, buildings, construction.
  Oxford university press.
- Maier, I., Rompf, T., and Odersky, M. (2010).
  Deprecating the observer pattern.
  Technical report.
- Wikipedia (2021).
  Observer pattern.