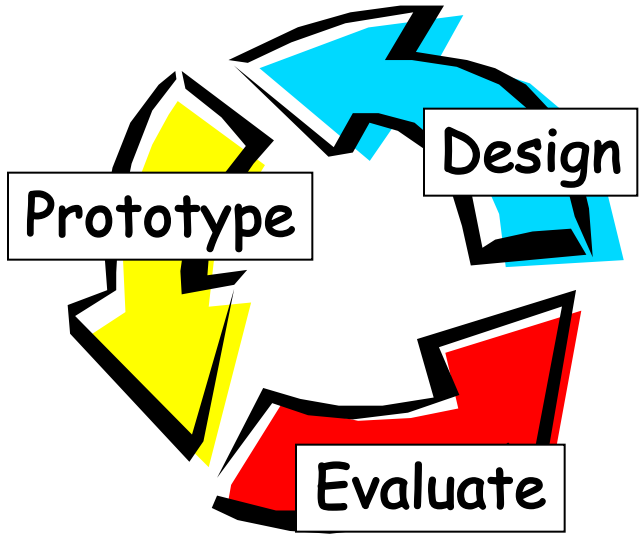
CS 338: Graphical User Interfaces

Lecture 1-2:
Design & Implementation

Human-Computer Interaction

- Human-computer interaction is a discipline concerned with the design, implementation, and evaluation of interactive systems for human use and with the study of the major phenomena surrounding them.
- The HCI lifecycle is an iterative cycle that involves designing and evaluating with "users" as much as possible.



Building User Interfaces

- Much of this process is smart, good ol' software engineering
  - applicable to the GUI parts of the system
  - applicable to the non-GUI parts of the system
- This course focuses on the implementation of the GUI parts
- BUT... We always keep in mind the other aspects of the process, particularly design

Mental & Implementation Models

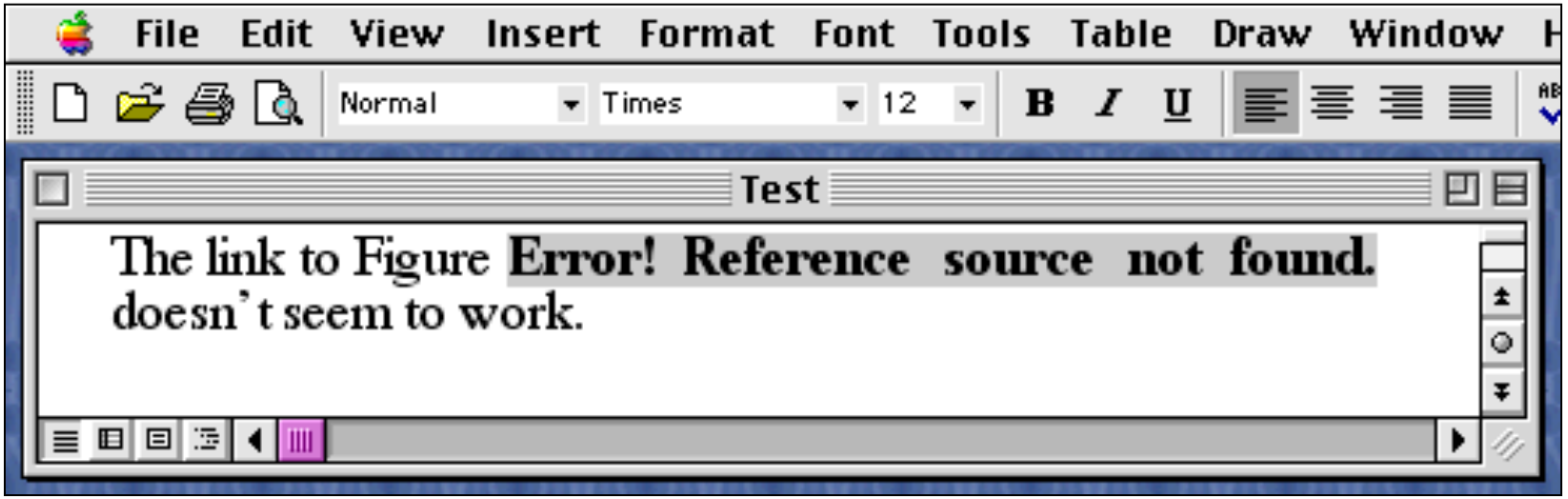
- Implementation models describe how *we*, as programmers, think about the system
  - there are many levels to these models
  - e.g., when you write "x++", we increment *x*
  - but what happens in the processor? cache? ...
- Mental models describe how the *user* thinks about the system
  - *not* the same as the implementation model!
  - e.g., saving changes (Word file vs. file system)
  - e.g., TV and movie projection
  - e.g., household electricity

Designing for the Mental Model

- Computer Science people build systems.
  Not surprisingly, then, most systems conform to *implementation models*.
- Book example: Windows file system
  - drag file within drive (e.g., C->C drive)
    -> file moves to new location
  - drag file to another drive (e.g., C->D drive)
    -> file is *copied* to new location
  - why? implementation of file system
  - this puts the burden of understanding low-level file system details on the user!

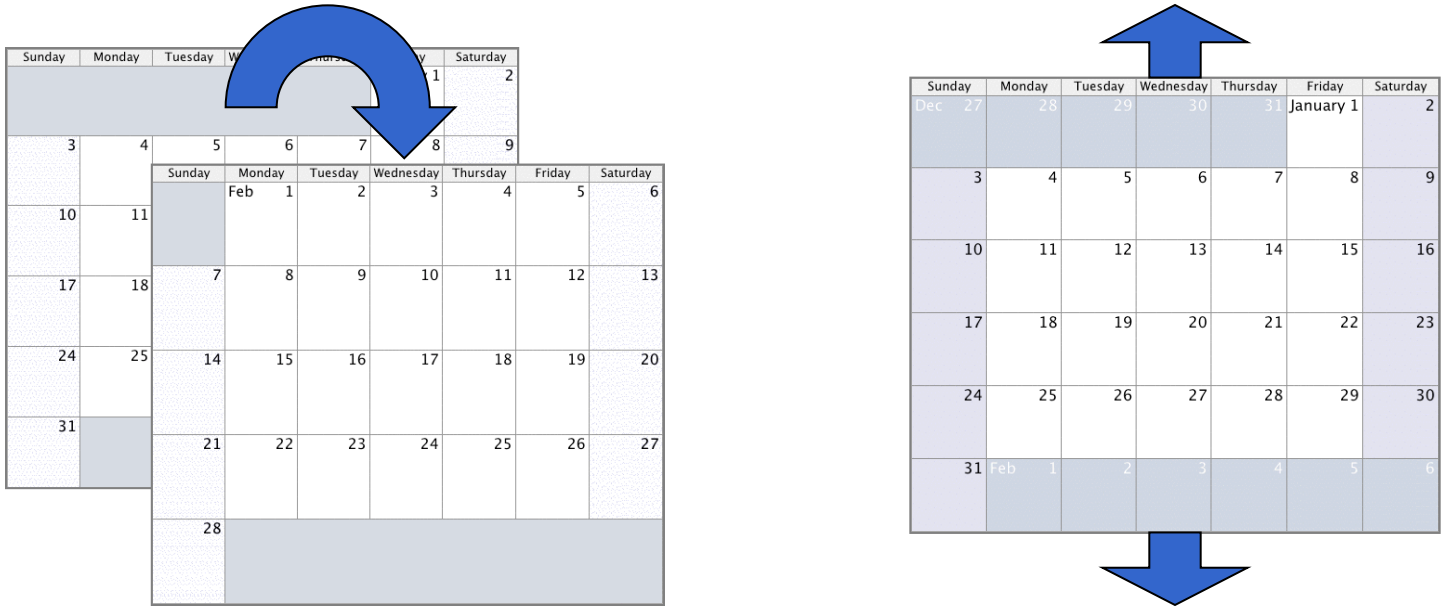
Designing for the Mental Model

- Another example: Word references
  - link number to other part of the document
  - mental model: this stays linked
  - implementation model: link gets broken, but software doesn't correct — makes you do it!



New Technology & Mental Models

- Sometimes, new technology warrants breaking old mental models — carefully
- Example: the good ol' calendar



"Significant change must be significantly better."

Design & Implementation: Summary

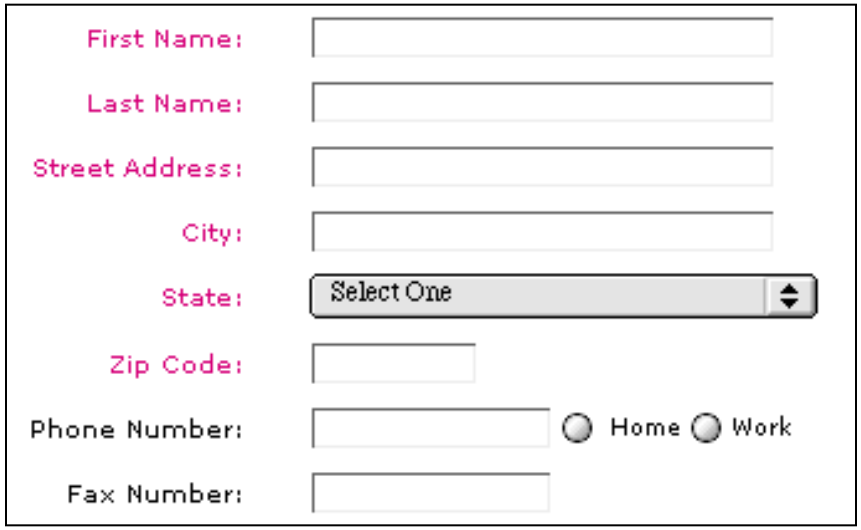
- Good, clean implementations facilitate good, clean user interfaces.
  - efficient code leads to faster response
  - well structured code leads to simple interfaces (and simple is good! -- easy to learn, remember)
  - robust code doesn't crash
- Good implementation doesn't make up for bad design, and good design doesn't help a bad implementation.
- Implementation & design go hand-in-hand!

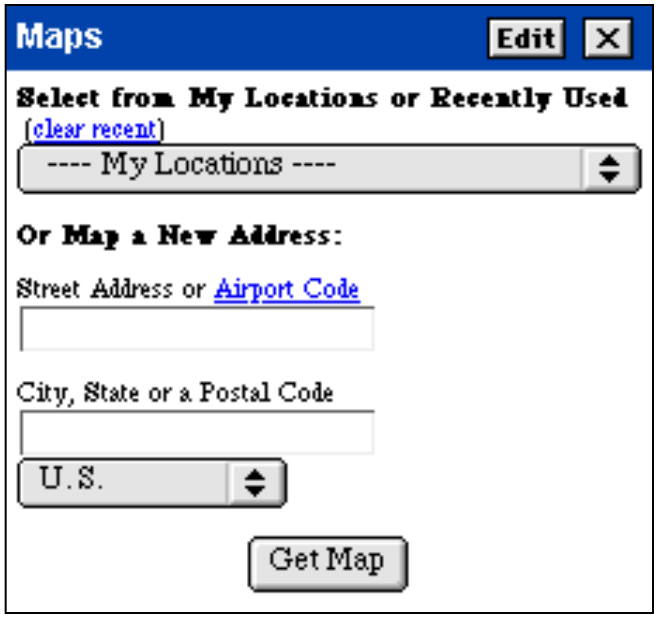
Design & Implementation Tradeoffs

- Example: Mac OS X launcher
  - design issues?
  - implementation issues?

Design & Implementation Tradeoffs

- Example: Two web-based input interfaces
  - design issues?
  - implementation issues?





When GUIs work well...

- Shorter training time -> lower costs
- Fewer errors -> less hassle, lower costs
- Less maintenance -> less hassle, lower costs
- High transfer -> lower future costs

- In other words:
  Good design saves time.
  Good design saves money.

Why are good GUIs hard to build?

- Multiprocessing: UIs are inherently concurrent
  - multiple inputs, redraws
  - synchronization, deadlock prevention
- Must deal with abort, undo, redo anytime (this requires a lot of state info to be kept)
- Real-time requirements
- Must be robust (users do lots of odd things)

Thanks to Scott MacKenzie @ York U for the next few slides!!

Why are good GUIs hard to build?

- API & UI logic complexity
- Reactive instead of proactive: the user dictates what the system should do
- Hard to modularize (OOP interface design)
- Exhaustive testing of UIs is hard – how to ensure robustness?
- Evaluation with users is time consuming
- And more...

Research Question of the Day

- We'll spend lots of time worrying about the user interface part of an application.
- What's the big deal? How important is this when implementing an application?
- Specifically, how much programming in a typical system is devoted to the interface?

Research Question of the Day

- Myers & Rosson conducted this survey
- They analyzed 74 systems of various flavors

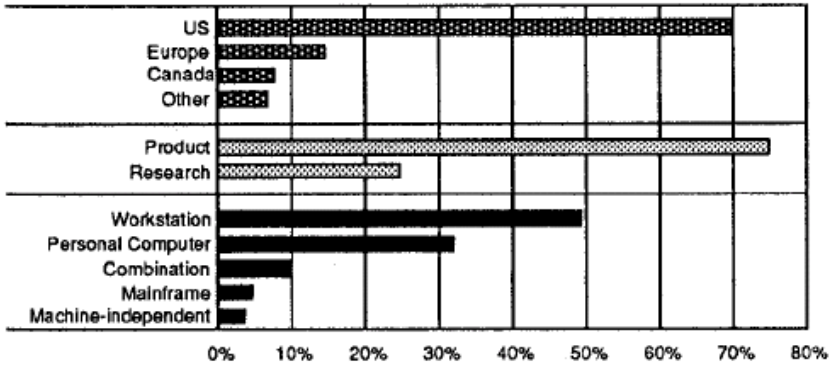


Figure 1:
Distribution of survey responses across countries, type of project, and host computer. The "Combination" systems used multiple types of computers at the same time. The "Machine-independent" systems were designed to run on different kinds of computers.

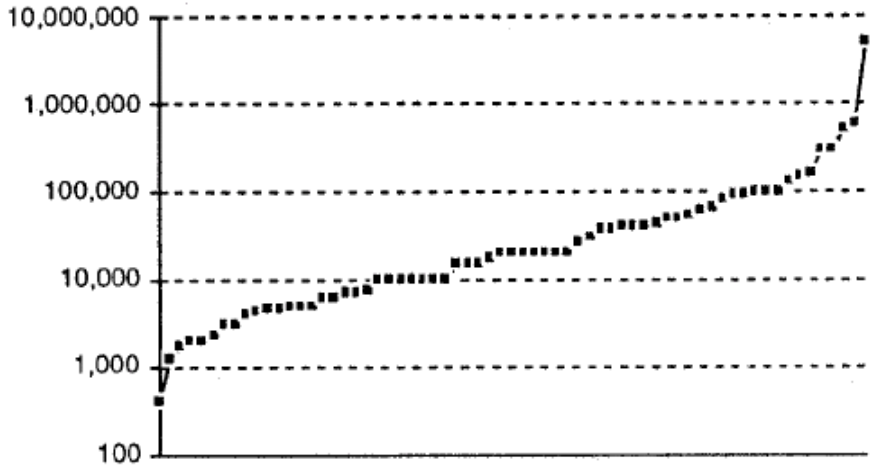
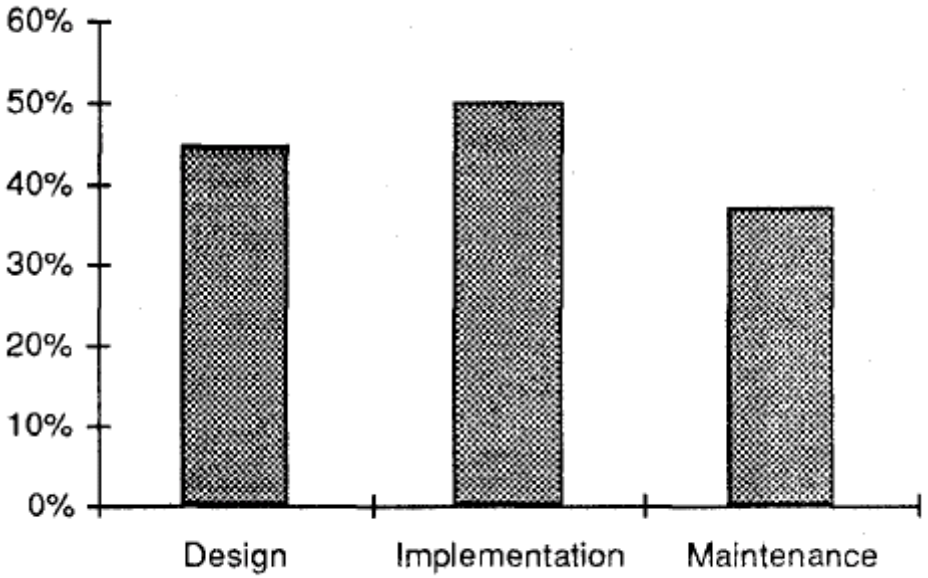


Figure 2:
Total number of lines of code on a log scale (for the 68 systems that reported a value).

Research Question of the Day

- Result: User interface required...
  - 44.8% of design time
  - 50.1% of implementation time
  - 37.0% of maintenance time



Research Question of the Day

- Interesting point: although UI = 50% of code on average, this varied greatly per system...

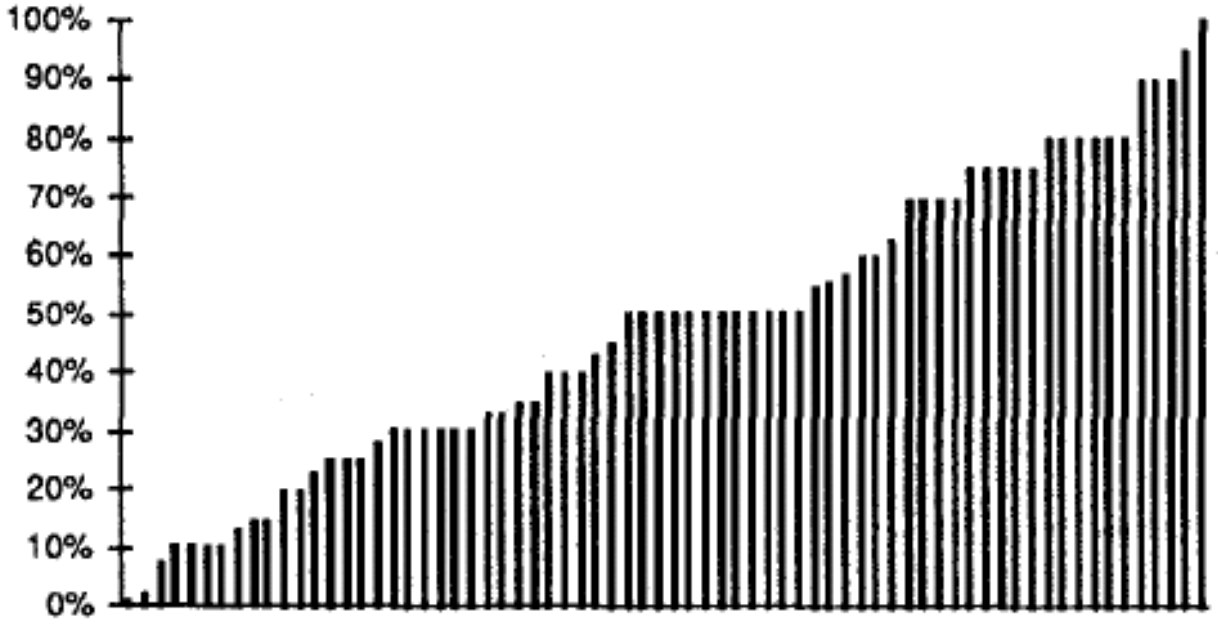


Figure 3:
The percent of the code devoted to the user interface (for the 71 systems that reported a value).

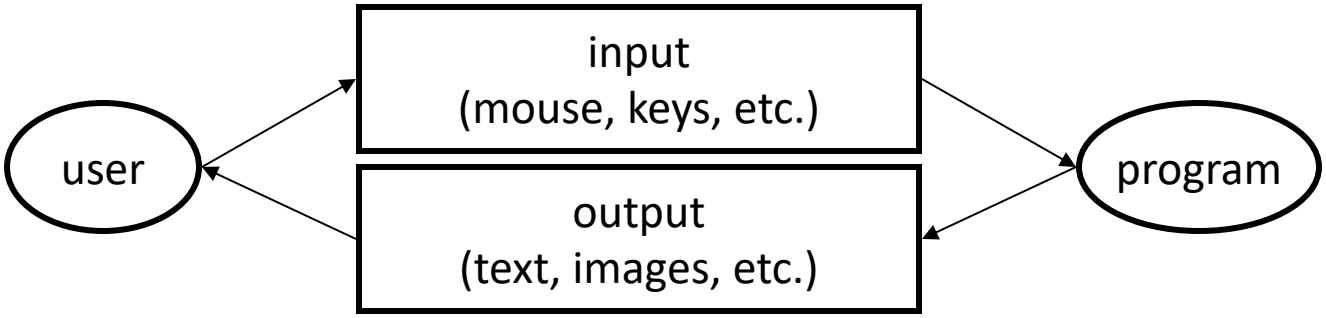
Research Question of the Day

- Other interesting points
  - prototypes & evaluation
    - 46% built running prototypes
    - 17% used paper-based designs
  - use of graphics
    - 70% 2D graphics, 14% 3D graphics
  - use of programming languages
    - many (>10) languages used
    - 58% used multiple languages in the same system

How do we implement GUIs?

- For many programs in CS coursework…

- For typical GUI applications…



- This leads to a very different style of design and implementation!

Implementation

- Our task for this course, generally speaking:

  Learn how to build great GUIs
  with a user-centered design.

- How can we build great GUIs?
  - There are A LOT of languages and technologies…
  - and A LOT of frameworks and components…
  - and normally you'll choose not just 1 or 2 options, but 5-10 that will all co-exist!

Implementation

Implementation

- Desktop apps
  - Common languages with standard GUI frameworks

Swing

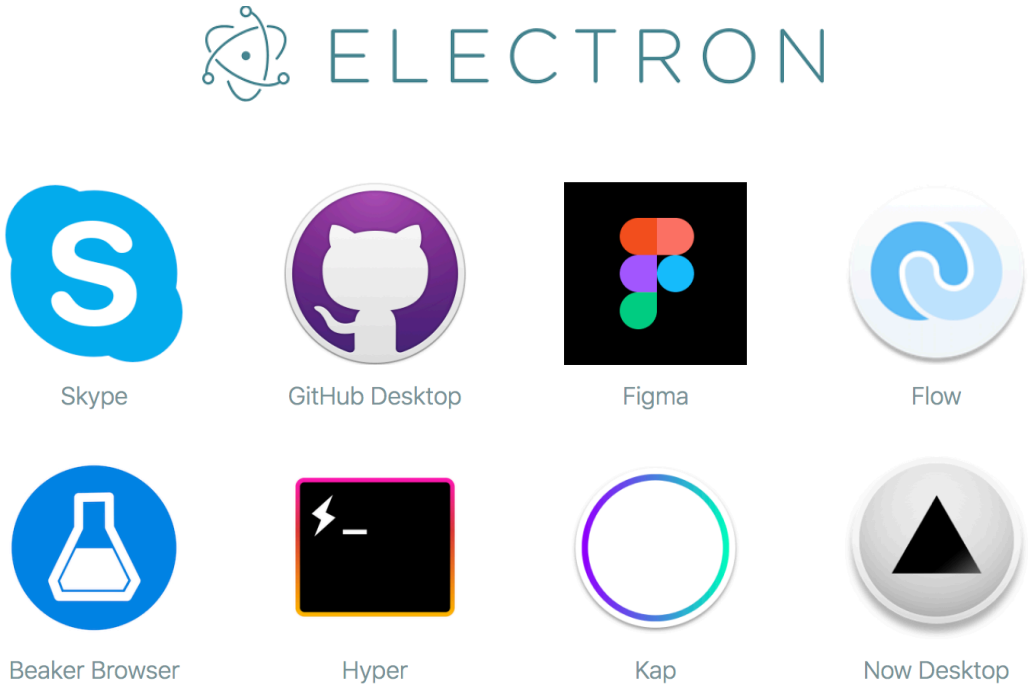
Tkinter

Implementation

- Desktop apps
  - Frameworks/languages primarily associated with a particular platform

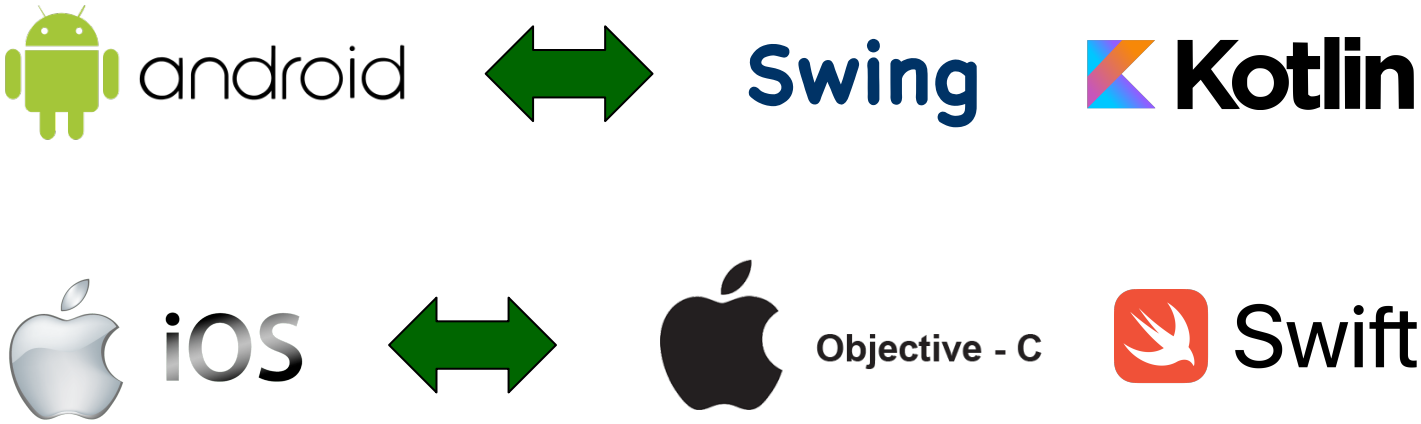
Implementation

- Desktop apps
  - More recently, web apps packaged as desktop apps



Implementation

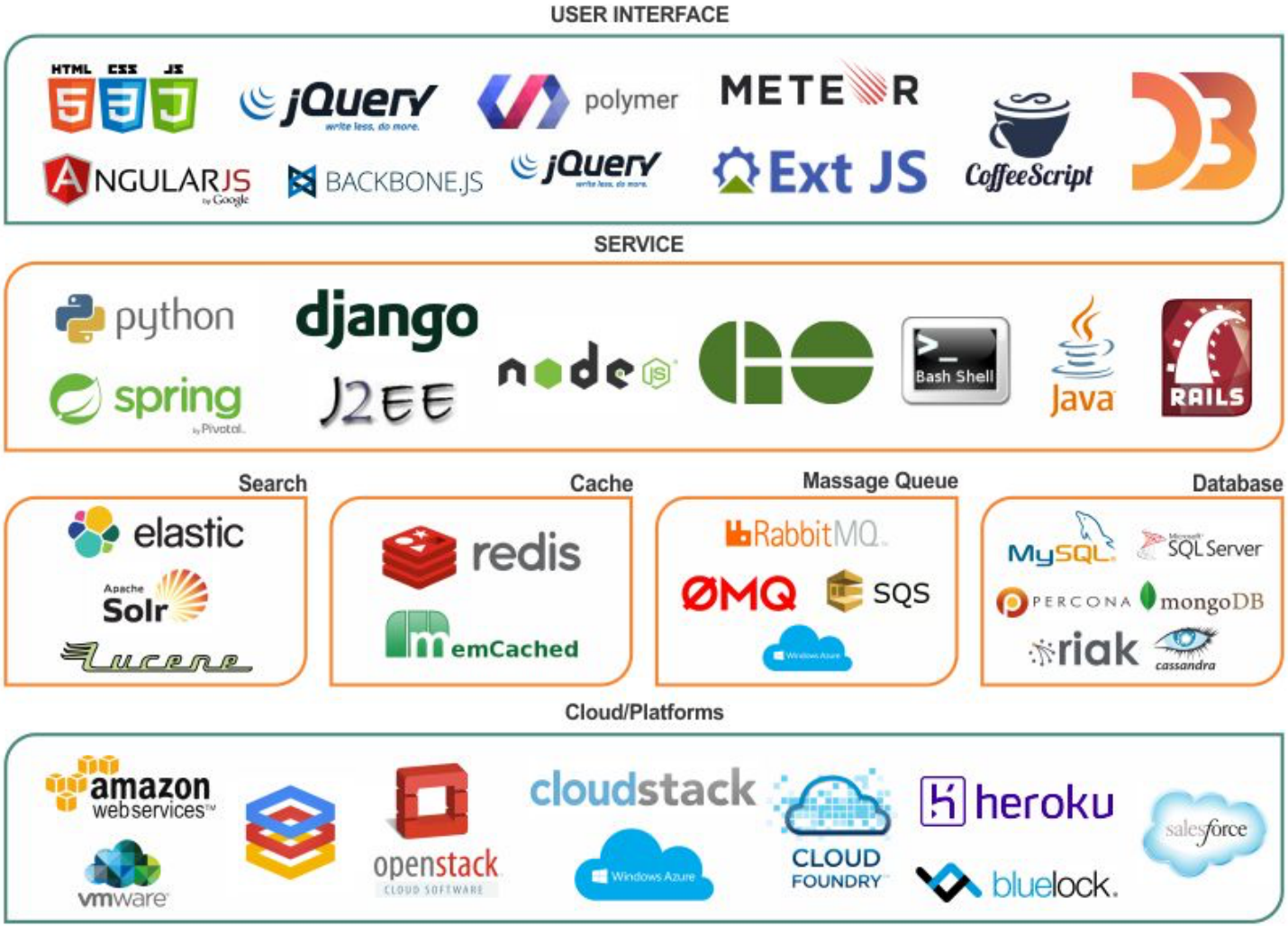
- Mobile apps
  - Languages/frameworks primarily associated with a particular platform



Implementation

- Web apps
  - Accessed via desktop or mobile browsers
  - More or less universal: everything needs web presence
  - Generally platform-independent
    - and most are “responsive” — adjust automatically to screen size
  - Even desktop apps are moving to web apps
    - e.g., Google docs/sheets/etc., MS Office cloud apps
    - (not to mention web apps embedded as desktop apps)
- Because of these advantages, web apps will be the focus of our implementation work
  - Though we will touch on all of these areas at some point

Full-Stack Development

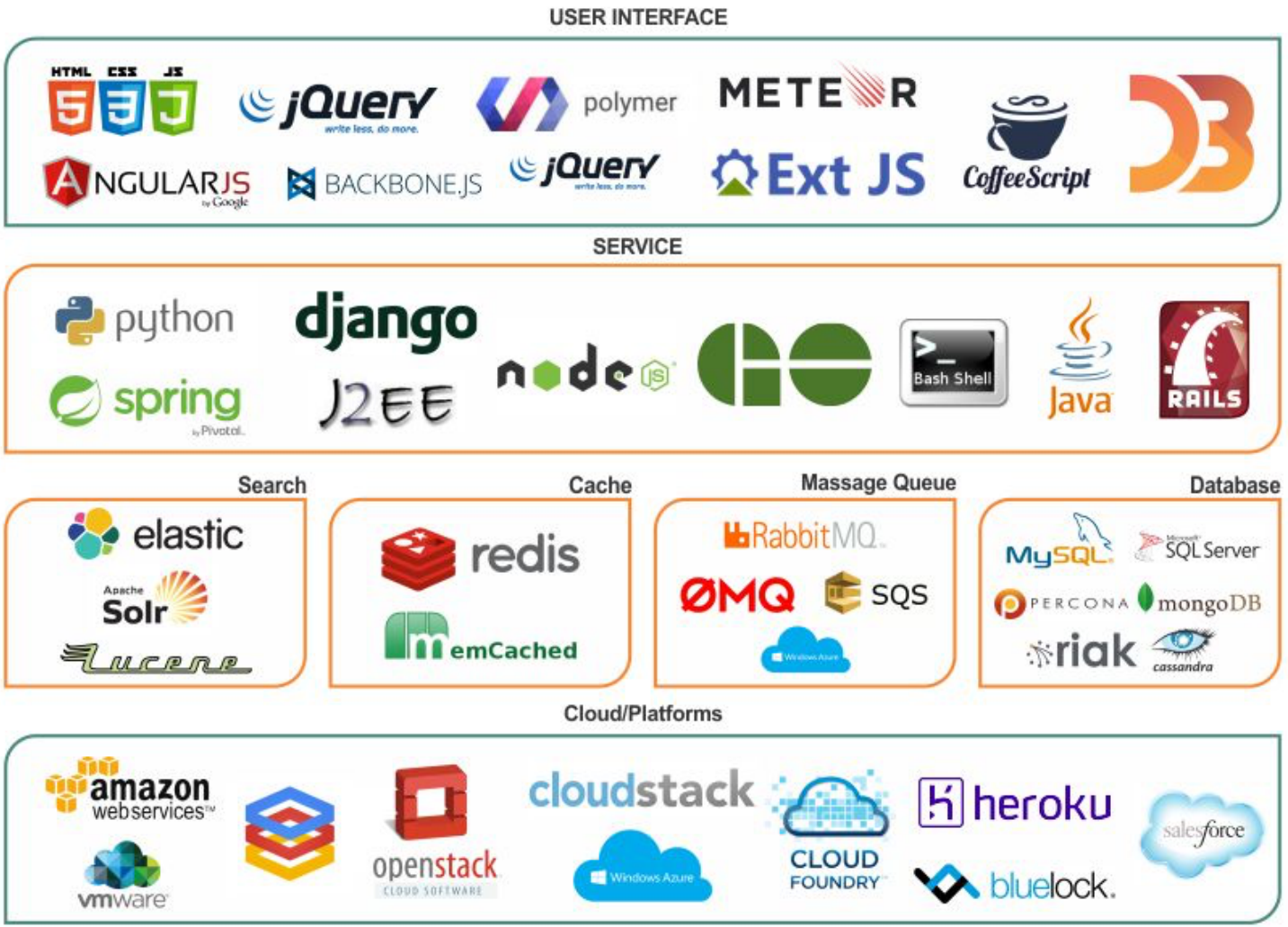


from http://www.byteorigin.com

Full-Stack Development

Our Focus

But we need these too!



from http://www.byteorigin.com

Our Stack…

- User Interface: **HTML, CSS, JavaScript, jQuery**
  - The essential components of web development
- Styling/Interaction: **Bootstrap**
  - Concepts transfer well to others (e.g., Pure)
- Web Framework: **Flask**
  - Simpler than many alternatives, but still very powerful
  - Concepts transfer well to others (e.g., Express, Django)
- Database: **SQLite**
  - Simpler than many alternatives, but still very powerful
  - Concepts transfer well to others (e.g., MySQL, etc.)
- Platform: **sandbox.cci.drexel.edu**

Course Overview

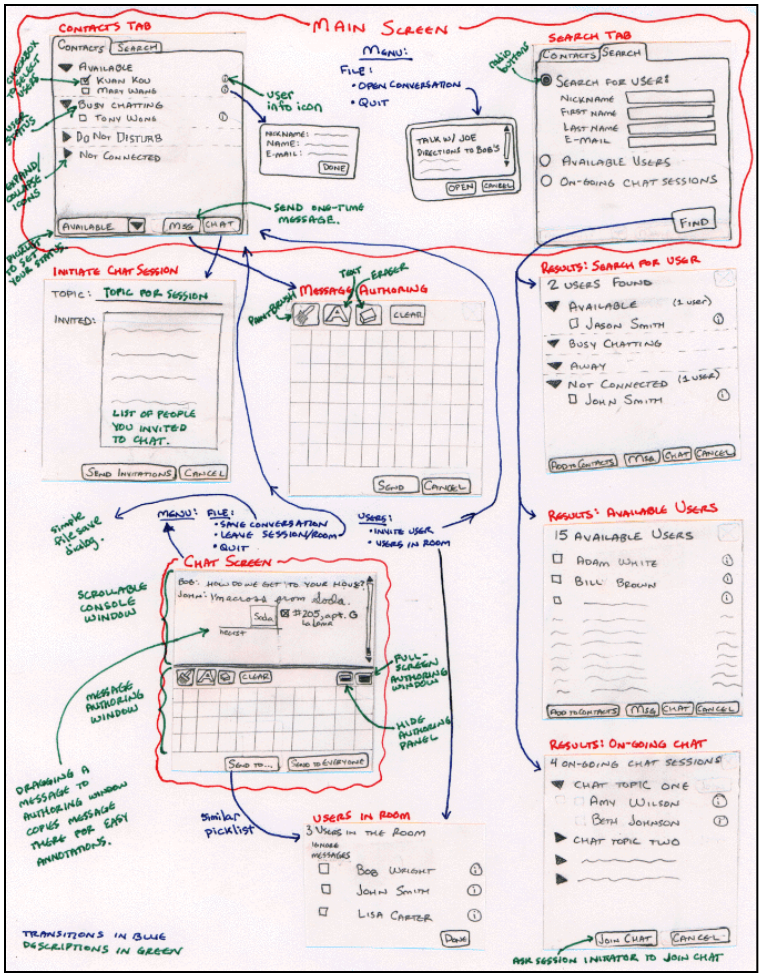
- We will study theoretical ideas of HCI alongside practical app development
- Most classes will start with a theory component, and then transition to an implementation component

Course Overview

- Week 2: Web apps
  - Put on your coding hats
  - Into the vast world of web-app development
  - Assignment #1 will be distributed at the same time, due 2 weeks later

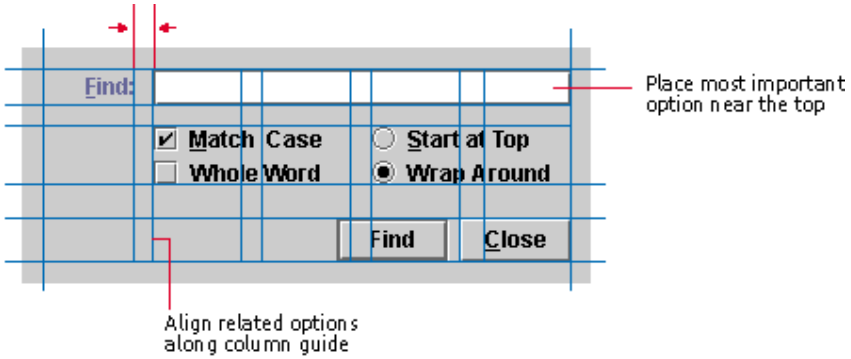
Course Overview

- Week 3: Prototyping and Graphics
  - Rapid sketching of a new interface
    - Used soon after for project prototype
  - Graphics basics
    - Underlying component of any GUI
    - Can do basic drawing with Swing itself
    - Other packages available for more complex graphics (2D, 3D)



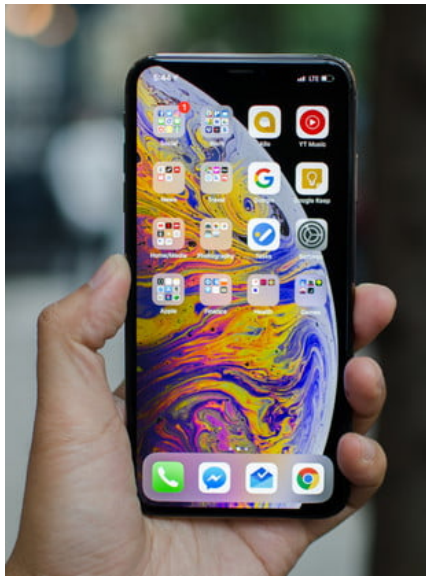
Course Overview

- Week 4: Evaluation & Guidelines
  - Brief introduction to evaluating a user interface (versus an entire course…)
  - Guidelines for “proper” design
    - We won’t delve into media arts-ish design (font choice, color scheme, subtle visuals, etc.)
    - We focus on CS-ish construction of a reasonable layout as dictated by Apple, Swing, and other guidelines



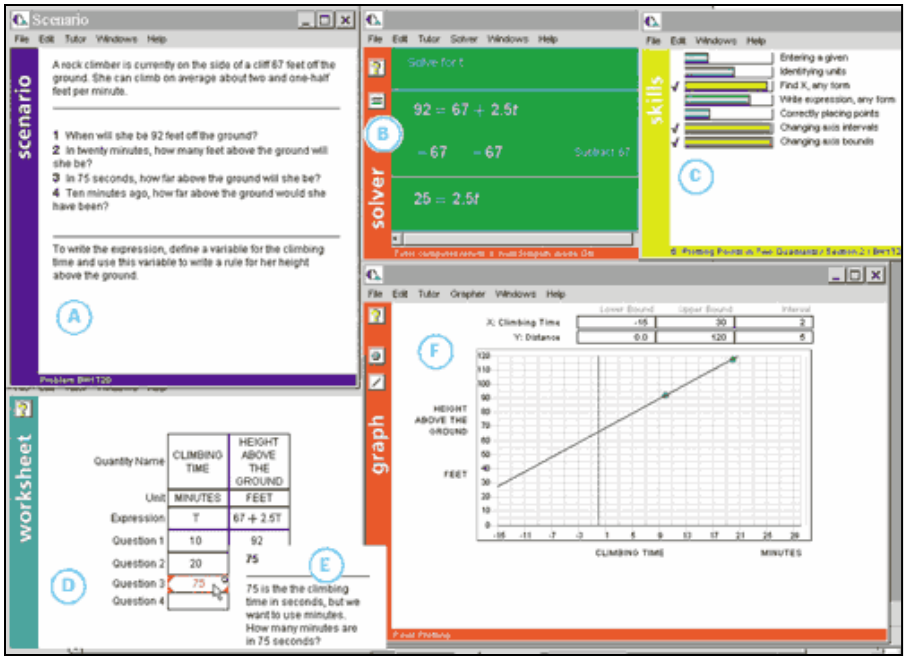
Course Overview

- Week 5: Advanced Input & Mobile Basics
  - Advanced input = anything beyond keyboard/mouse
    - Or another way to look at it: how “intelligence” enables different kinds of input
  - [Maybe] Mobile programming
    - A brief look at mobile (iphone) interface programming
    - Honestly, I’m no expert (any experts here?)
    - Overview of the basics



Course Overview

- Week 6: Intelligence & Beyond WIMP
  - Building intelligence into user interfaces in a modern way: cognitive models
  - Going beyond WIMP (windows, icons, menus, pointers) interfaces

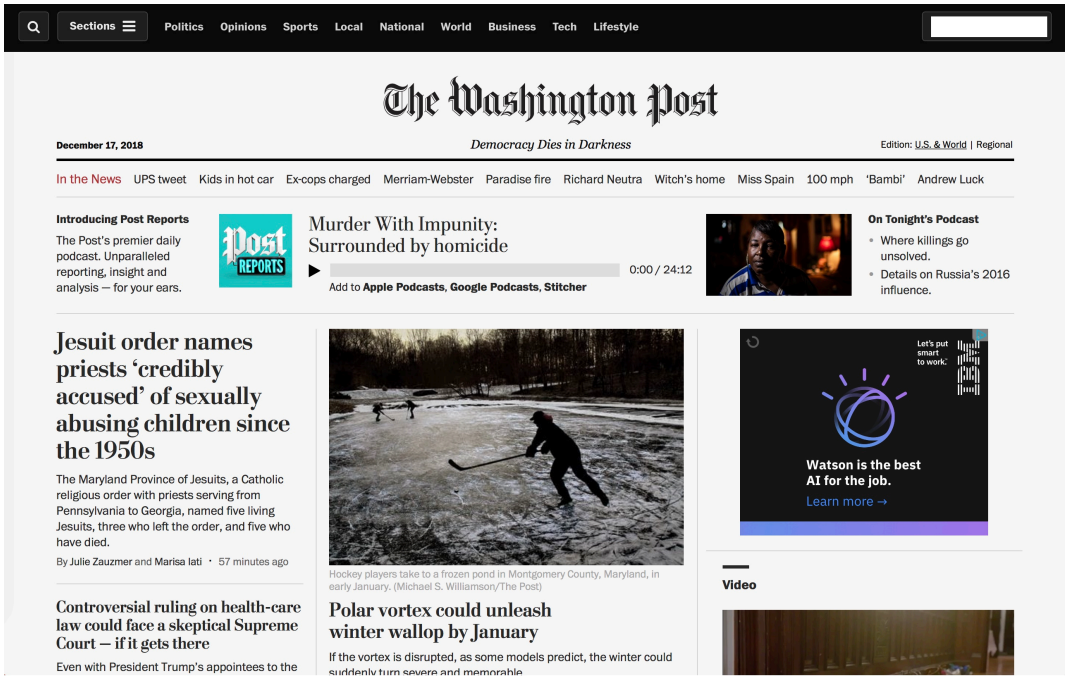


Course Overview

- Week 7: Assistive Technology
  - User interfaces that help people with disabilities
  - Technology involved
    - Sometimes very low-tech
    - Sometimes very high-tech

Course Overview

- Week 8: Interfaces on the Web
  - How web pages ?= user interfaces
  - Web dev extras
  - Java applets



Course Overview

- Week 9: Interfaces off the Desktop
  - Beyond web-ish uses on the desktop
  - For example:
    - Handhelds
    - Kiosks
    - In-vehicle interfaces
    - Ubiquitous computing

Course Overview

- Week 10: Mini-Conference
  - Your projects!
  - On-campus students: short presentations + demos
  - Online students: online presentations + videos when possible

For Next Week…

- Finish Lab 0
  - Ungraded