Social Computing Systems

Walter S. Lasecki

EECS 498, Winter 2017
(http://tiny.cc/socsClass)
Today

- On what platforms does Social Computing occur?
- What is in our ‘toolbox’ for system building?
Platforms
‘Desktop’ Computing
‘Desktop’ Computing
‘Desktop’ Computing
‘Desktop’ Computing
‘Desktop’ Computing
‘Desktop’ Computing

- More local computational power
- Larger interaction surfaces (e.g., touch screens, monitor space, etc.)
- More stable internet connection

- Future Advances:
  - Even more computational power
  - More affordable, and thus available to new populations
  - New interaction modalities
Mobile Computing
Mobile Computing

- Wireless
- Portable
- Always (or often) -available

- Future Advances:
  - Lighter
  - Smaller (?)
  - Better battery life
  - Reconfigurable (e.g., Google’s Project Ara)
  - New attributes: flexible, brighter, including …
Wearable Computing
Wearable Computing

- Enables on-body interaction
- Always-'on’ (let’s assume)

- Future Advances:
  - Lighter, smaller
  - More computational power
  - More integration with existing devices, such as…
VR / AR
Projected Interfaces
Ubiquitous Computing (UbiComp)
Ubiquitous Computing (UbiComp)

- Devices / Gizmos / ‘Internet of Things’
- Distributed networks of devices and sensors
- Focused purposes for connected devices

- Future Advances:
  - Smaller, less obtrusive (allowing them to be added to more devices)
  - More affordable
  - Better sensing and power efficiency
  - Better use cases and interpretation algorithms
Custom Devices
Custom Devices

- More design freedom
- Expensive to produce (often requires tooling)
- Expensive to buy (as a result of production cost)
- Barrier to adoption
  - Sometimes can help too — Apple achieved ‘status symbol’ status

- Future Advances:
  - Downloadable, thanks for 3D printing
  - Reconfigurable
Platform Recap

- Things (devices)
- Links between things (networks)
- How we build software on things (frameworks)
- Systems (← our job)
Platform Recap

- Desktop (in a fixed location)
- Mobile (wireless)
- Wearable (on-body)
- Ubiquitous (all around us)
- Custom (special-purpose hardware)
Supporting Technology
Network Infrastructure

- The ‘Internet’: network of connected computers/servers
- The ‘Web’: protocol layer and standards on top of the internet
- WANs: Wide area networks — e.g., company (or university) networks
- LANs: Local area networks — e.g., home networks
- PANs: Personal area networks — e.g., network of mobile/wearable devices
Tools / Frameworks / Packages / Languages

- **Languages**
  - e.g., Javascript, C++, Ruby, etc.

- **Tools/Libraries**
  - Java Standard Lib, scikit-learn (Python)

- **Frameworks**
  - MeteorJS, [Ruby on] Rails

There are many MANY more examples out there — look for the right one for a project!
Application Programming Interfaces (APIs)

- Implements a common communication language among code
- Abstract away internal changes from 3rd party developers
- Most/all major social computing platforms have API access
  - Facebook, Yelp, Twitter, Uber, … what APIs have you used?
- Lets us more easily use other people’s SoCS as platforms
  - Build new feature → deploy to existing user bases
Examples: Web Front-End Tools

- JavaScript (lang)
- CSS (styling)
- HTML (lang)
- Handlebars / Moustache (MVC templating)
- ReactJS (MVC framework)
- AngularJS (MVC framework)
  - “AngularJS is what HTML would have been, had it been designed for building web-apps.”
Examples: Web Back-End Tools

- PHP (language)
- Ruby on Rails (framework)
- Apache (webserver)
- NGINX
- Node.js
- MeteorJS
- ...

Example: Web Storage Tools (databases)

- MySQL (centralized)
- SQLite (localized)
- MongoDB
- SQL Server (MS)
- DB2 (Oracle)
- ...
Examples: Packaged Web Support

- Heroku (hosting + frameworks)
- Firebase (back-end)
- ...


We need more examples!

- Wearables (e.g., Google Watch ADK)
- UbiComp
- ...
What is the difference between a Framework, a Toolkit and a Library?

The most important difference, and in fact the defining difference between a library and a framework is Inversion of Control.

What does this mean? Well, it means that when you call a library, you are in control. But with a framework, the control is inverted: the framework calls you. (This is called the Hollywood Principle: Don't call Us, We'll call You.) This is pretty much the definition of a framework. If it doesn't have Inversion of Control, it's not a framework. (I'm looking at you, .NET!)

Basically, all the control flow is already in the framework, and there's just a bunch of predefined white spots that you can fill out with your code.

A library on the other hand is a collection of functionality that you can call.