CSE 403

Architecture, Part III

Note new time

Startups...Seattle...You?

October 22, 2012
6:00PM
Room: EEB125

Free Food
Door prizes ($100 Amazon Gift Cards)
Scholarship opportunities
Bring a resume (if interested)

Hear first hand from successful leaders of the startup world.

Questions? Contact yamamoto@cs.washington.edu

Presentations

- Startups: Why you?
- 5 tips for effective startup engineers
- Show me the money: A view from venture capital
- Why Seattle is a great place to startup
- Counterpoint: Why not Seattle?
- Startups: A great place to intern

Announcements

- You'll get feedback on your preliminary Ar
  Friday morning.
- This week you are makin
  etc. More depth ar
  flows comple
  unders
  sc
  major ux
  omplete,
  all features, proposed
  ss designs, starting
  22
  ith milestones and proposed features and a
  prototype" as well
  w requirement -- a project metric
  ject presentations on 10/24

Think: Why are you doing what you are doing?

- **Are you specifying useful requirements?**
- ~~Are you just writing a requirements doc?~~
- ~~Are just trying to satisfy a class assignment?~~

Project metric

- A metric is a **quantitative** measurement of how well something (your project) is going.
- You must decide on 1 metric to measure your project
- You must include this in your requirements document: with dates when you will achieve milestones
- Why one?
  - Focus
  - You can only do one thing well
  - If you must use 2, we'll allow it. But it probably means you aren't focussed enough

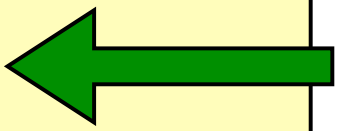
Presentations (10/24)

- Update on where you are at: Requirements and Architecture, et al
- 7 minutes
- One individual or team effort, your call
- Presentation order randomly selected, using same format as proposal presentations
- Send pdf of presentation by 11:59PM, 10/23 via dropbox

Presentation format

- 1 minute - Elevator pitch: at least the1 sentence summary and your metric
- 1 minute - Organization: what, names, roles, why (a picture would be good)
- 1 minute - Technology choices/Architecture: what, why, potential alternatives
- 1 minute - Process: what and why
- 1 minute - Milestone/Proposed schedule (what you propose to deliver at 11/5, 11/19, 12/3)

- 1 minute: Issues: What puts your project at risk
- 1 minute: Something interesting you've learned
- 1 minute: Quick demo of prototype



Choose 1-3

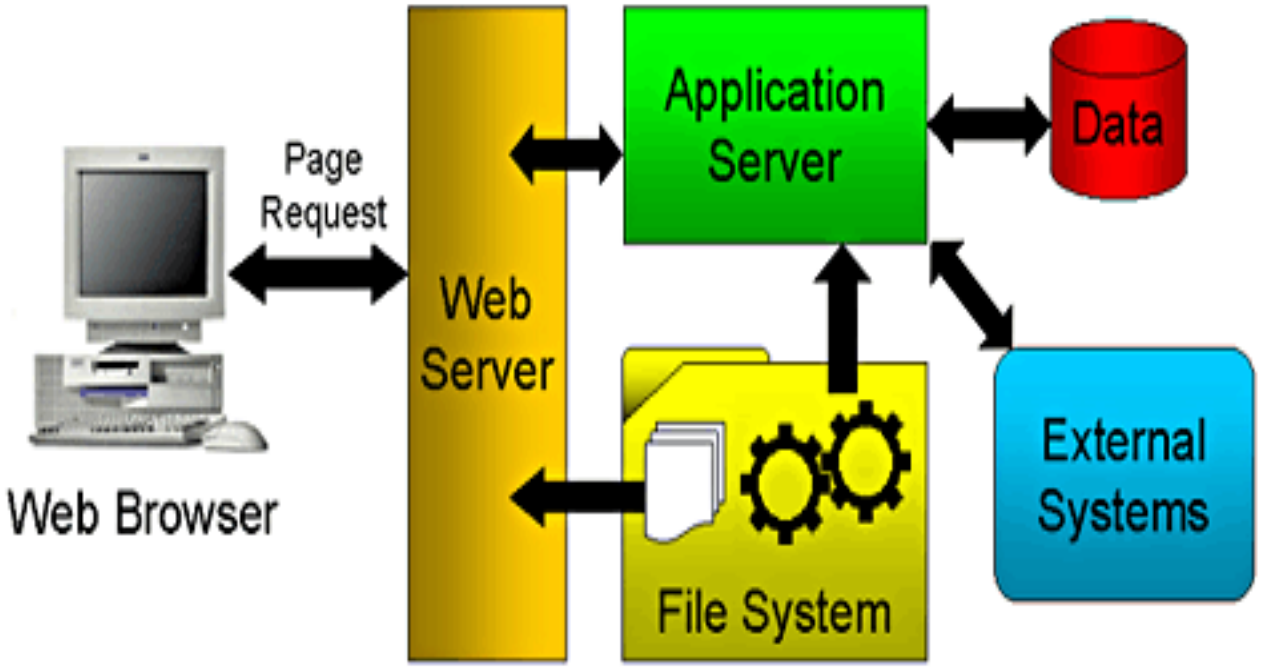
Late policy

- (Revised) project docs due 10/22: we'll grade what you have in place, so "no penalty"
- Assignments: 100% penalty without good excuse
- Missed project milestones: Evaluated on a one-off basis, largely determined by plan for correction (your presentation is part of the milestone grade)
- Missed final writeup: 10% off for each calendar day late

Major themes from Friday

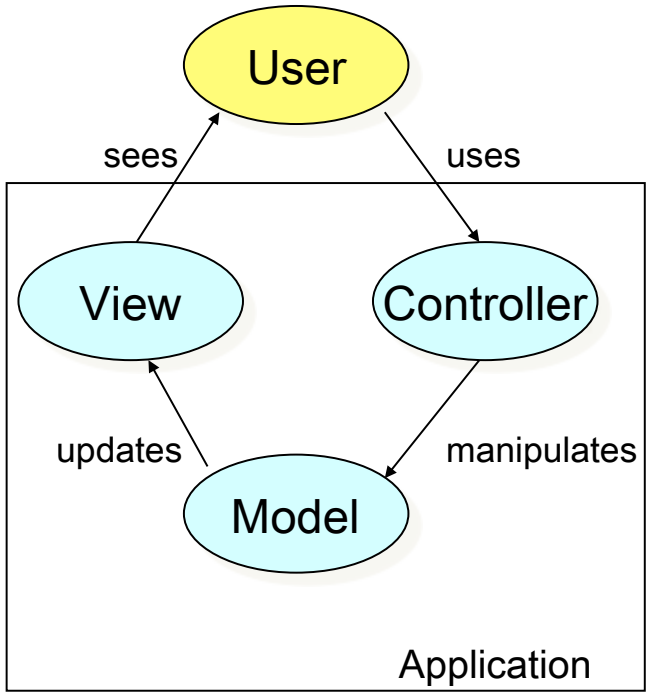
- Divide and conquer
- Modules: Interface and Implementation
- Coupling (Loose is usually good across module boundaries)
- Cohesion (Strong is usually good within module boundaries)

Traditional (old school) web application architectures usually look like this



Some common architecture patterns

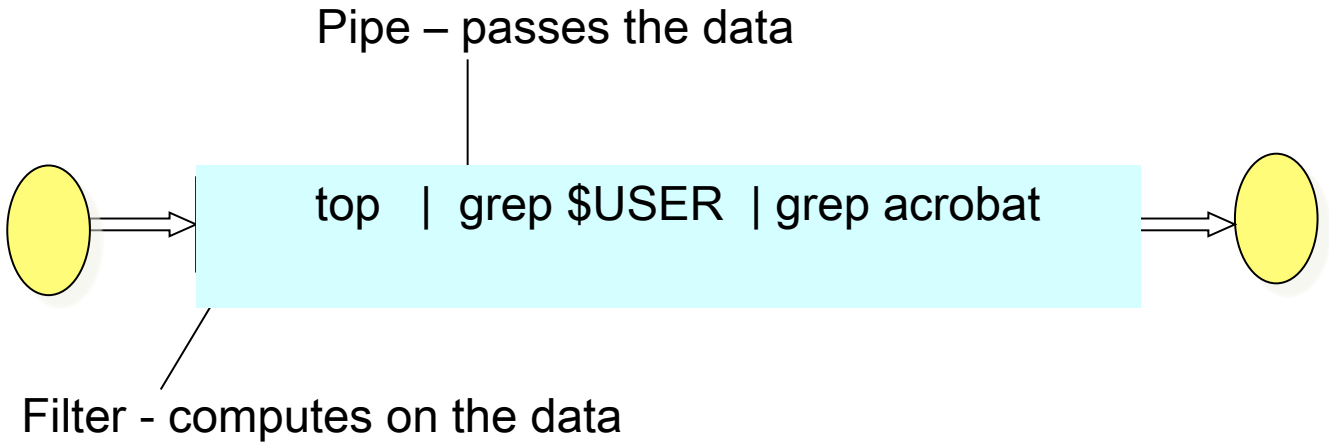
Model-View-Controller



Separates:

- the application object (model)
- the way it is represented to the user (view)
- the way in which the user controls it (controller)

Pipe and Filter

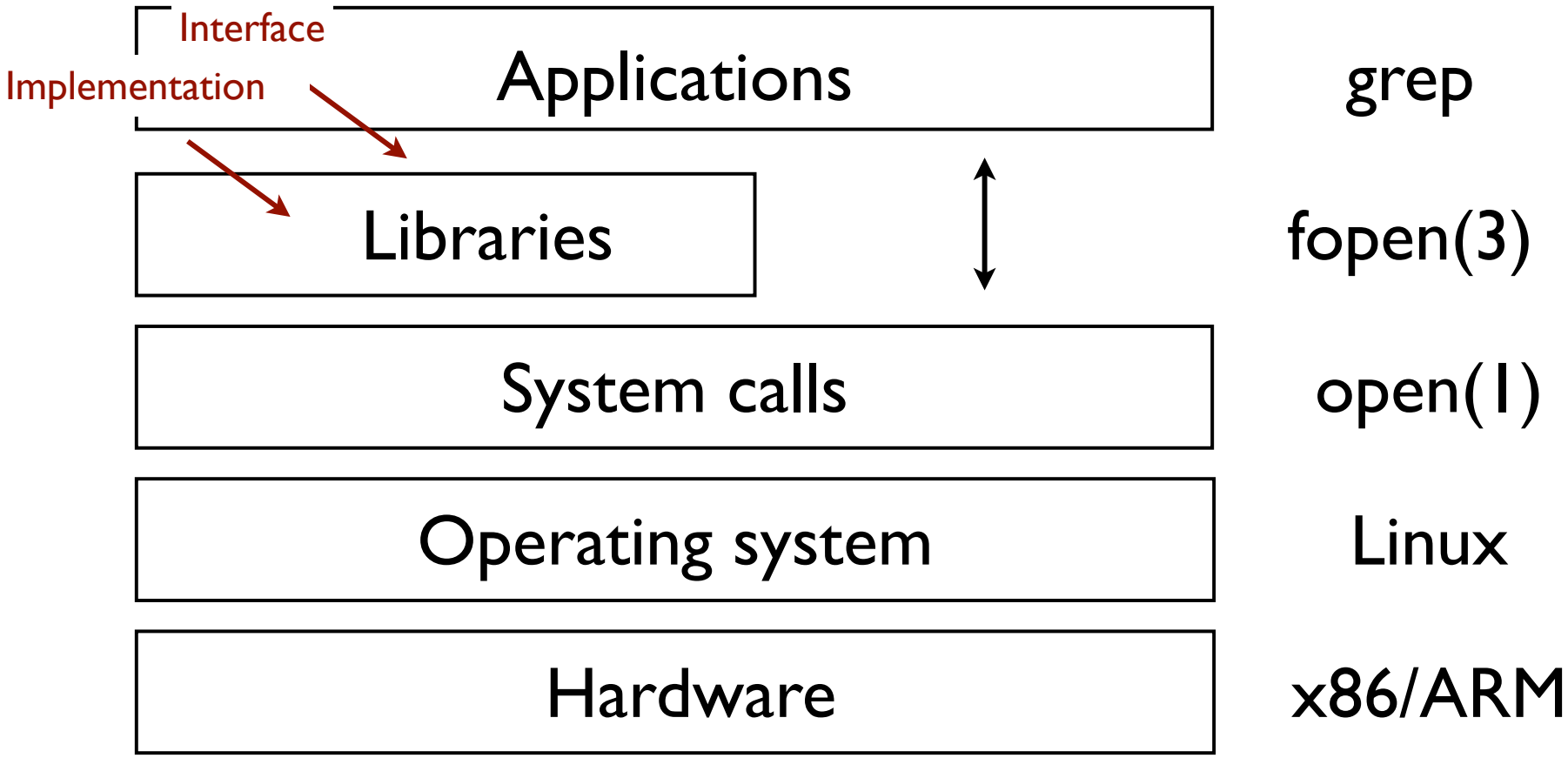


- Each stage of the pipeline acts independently of the others

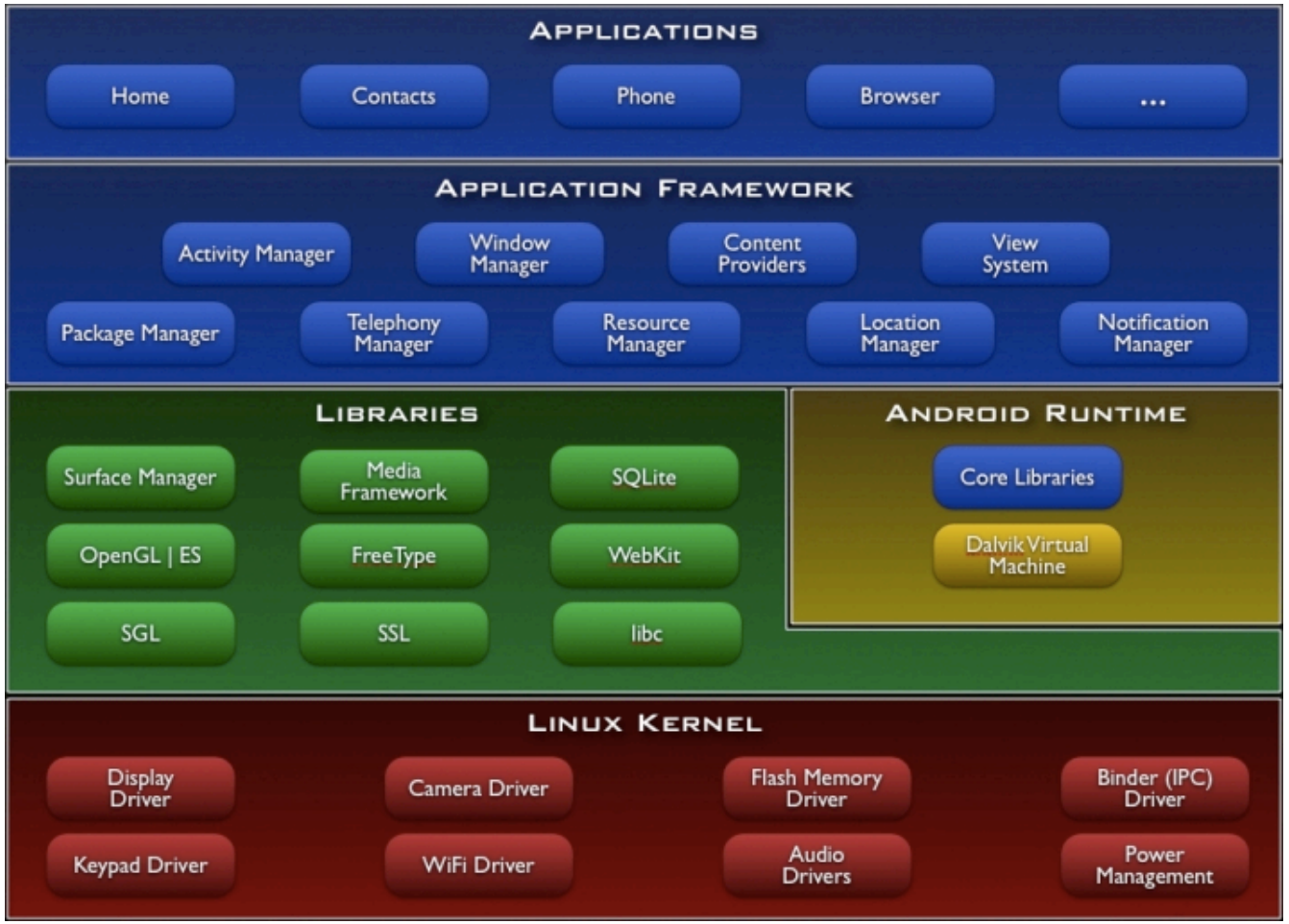
- Can you think of a system based on this architecture?

Architecture Modularity: "Layer Cake" Abstraction

Layers of abstraction and implementation

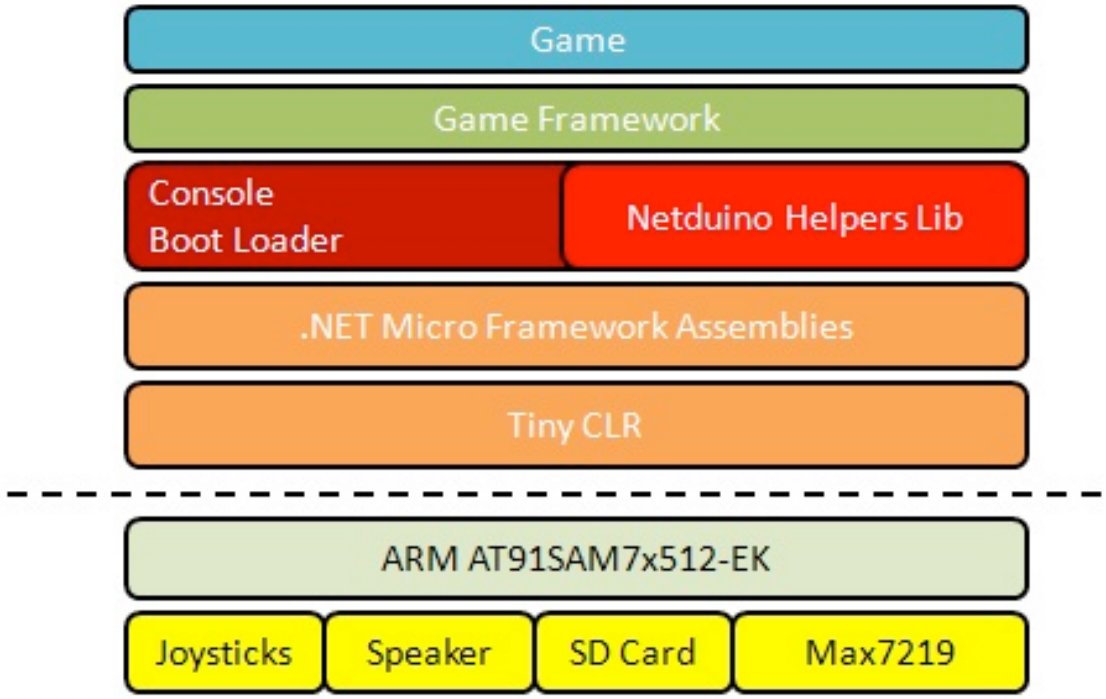


Android

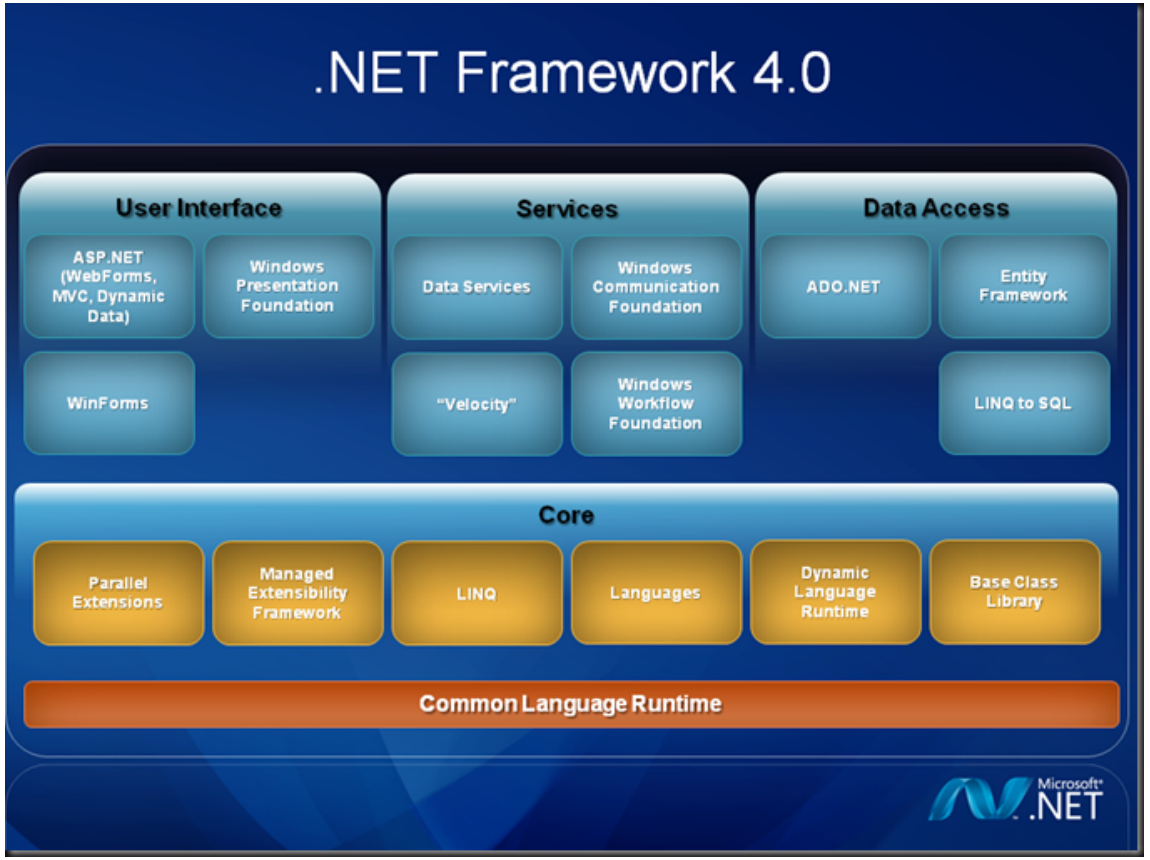

Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Game console

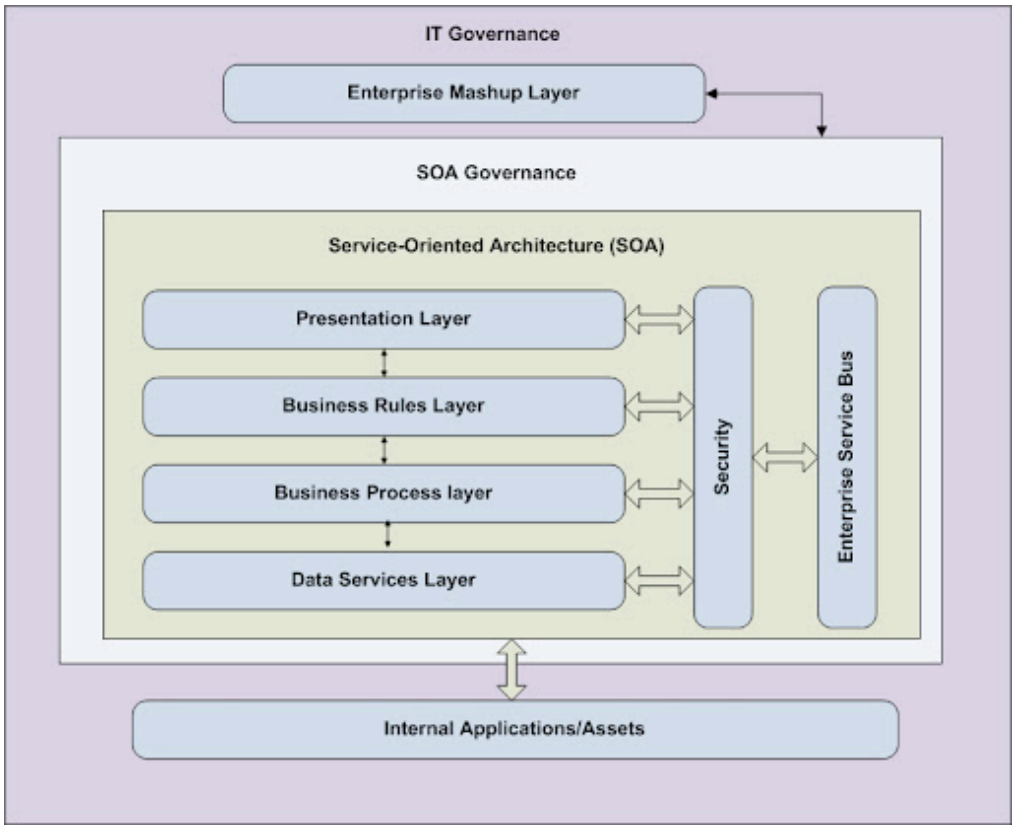
The Console Software



.NET Framework



Service Oriented Architecture



Philosophies ("religious" wars)

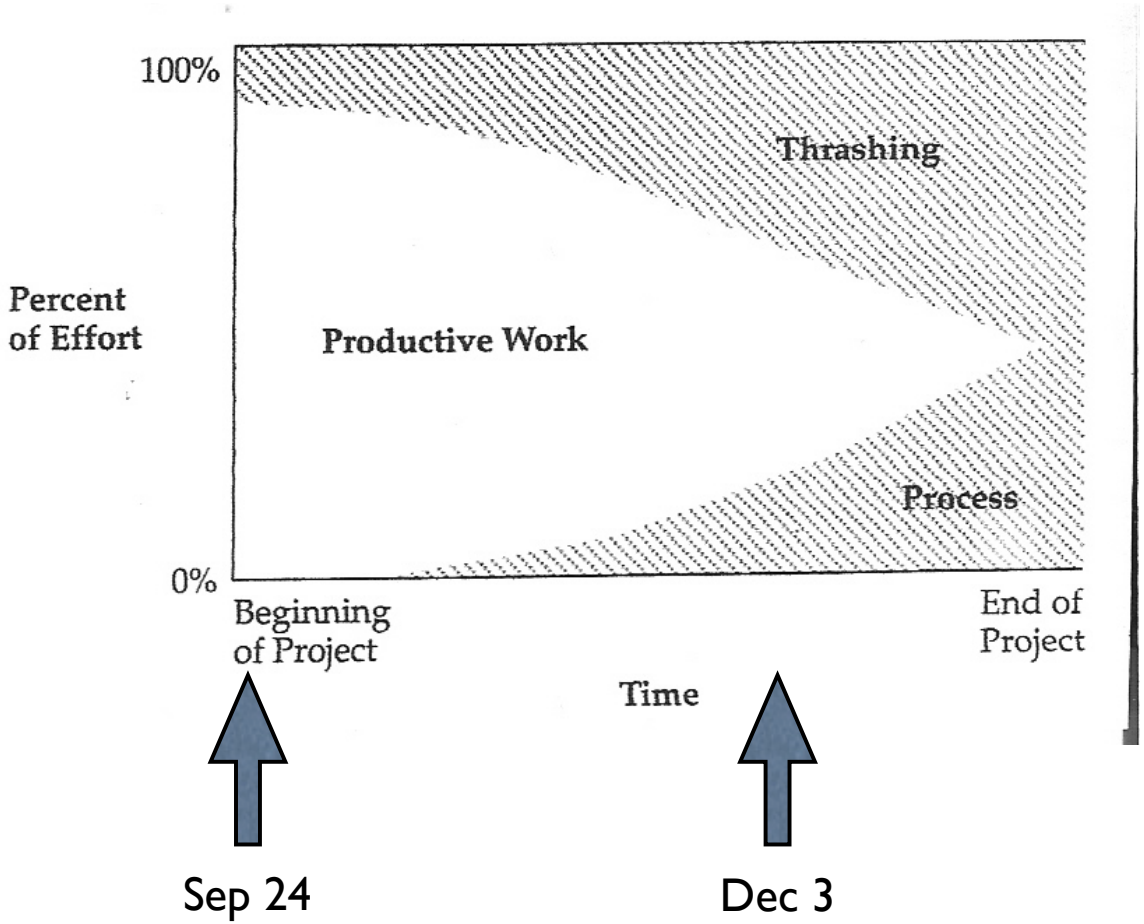
Philosophies ("religious" wars)

- Programming languages: "strong"/compile time typing vs. "dynamic"/runtime typing
- Data stores: RDBMS vs. nosql?
- Native apps vs. HTML5?
- How strong should your beliefs be?

What does this mean for your project?

- If you are having architecture problems, you are doing something tremendously right or horrifically wrong
- Except: Nuts and bolts from CSE331/CSE332 are important
- Except: You must "architect" your organization, tools/runtime, and processes
- Your project may be highly functional and you might not worry much about other aspects of architecture (e.g. the "-ilities")
- ...but disaster may strike and you are in trouble (just like the real world)

What's the trade off for your project? (process/infrastructure vs. "real" work)



Process/Infrastructure vs. real work guidelines

- Plan as though your project will have a lifetime of at least a year past 12/2012
- Plan to put your project on GitHub as part of your personal professional dossier in the public
- Be proud of your work!

Take aways

- Architecture -- the big picture, technically
- Tools and runtime
- Architecture applies to your product as well as your organization
- Primary goal is to get stuff implemented but the "-ilities" matter too
- Key concepts:
  - **Divide and conquer**
  - **Abstraction is your friend**
  - **Modules: interfaces and implementation**
  - **Loose coupling is good/Strong cohesion is good**

Architecture, Design, Tools, Process (Team, Schedule)

- Architecture
  - Data/Information architecture, components and interfaces
  - UI/UX in requirements doc
- Tools and runtimes
  - Included physical topology
  - Rational for your choices
- Development process
- Schedule (add it requirements doc)
- Organization
- **Missing: testing, deployment....**