Software Engineering
ECE444

Introduction

Michael Stumm
ECE
University of Toronto

About the instructor

- Contact information
  - Email: stumm@eecg.toronto.edu (best way to contact me)
  - Office: SF2001
- Online presence
  - Webpage: none
  - Facebook: zero posts
  - LinkedIn: https://www.linkedin.com/in/stumm/
  - Twitter: @stummm
- Research Interests
  - Systems Software incl. operating systems
  - Software engineering
  - Publications: https://scholar.google.ca/citations?user=SfN31P4AAAAJ

Course Objectives

- Learn how software is developed in modern tech companies
- Learn by doing

Premise: Software is Important

- Computers are everywhere:

  Financial & insurance industry, government, hospitals, restaurants, stores, gas stations, cars, trains, planes, bombs, missiles, watches, pace makers, TV's, tablets, cameras, (smart) phones, etc….

  ➡ Software is everywhere

- Software key in every discipline
  - Electrical engineering, physics, chemistry, biology, medicine etc.
- Software key to business:
  - Differentiation: new unique features & capabilities
  - Competitiveness: cost-advantage
- Software: 3rd largest industry
  - TO a big hub

- Software is changing the world as we know it !

Yet most software sucks…

Typically Software…

- has lots of errors
- noticeably crashes periodically
- has poor user interface
- doesn't work properly for some features
- wastes resources
- is not secure
- doesn't perform well and is not scalable

An estimated 50% of the software produced
is never used because it is useless or doesn't work

Famous bugs I

- ObamaCare's Exchange (>$700 million)
- British NHS system scrapped after spending $20B
- US Air Forces cancels ERP project after spending $1B
- US FAA Air Traffic Control system scrapped after spending $2.6B
- IBM OS 360: 1960's

  Over 1 year late

  Shipped with 1000 known bugs only to increase

  200 Person Years to 5,000 Person Years
- IBM Workplace OS: 1999

  Cancelled after $2 billion investment
- Ariane 5 explosion: 1996

  Integer Overflow

Famous bugs II

- Canadian Therac-25 radiation machine burns 6 : 1985
  - Software never tested

- F16 flips over when crossing the equator: ’79

- San Francisco BART:
  - ghost train in tunnel
  - doors open while train at full speed

- Woman kills daughter and self
  - software bug incorrectly diagnoses incurable disease.

- North America Blackout of 2003

In fact…

- 28% of banking portals are secure
  16% of gov portals

- 32% of software projects are successful
  - 50% of them cost > 30% than expected
  - 35% very late

- 80% of dev costs:
  identifying and correcting defects

Why all these problems?

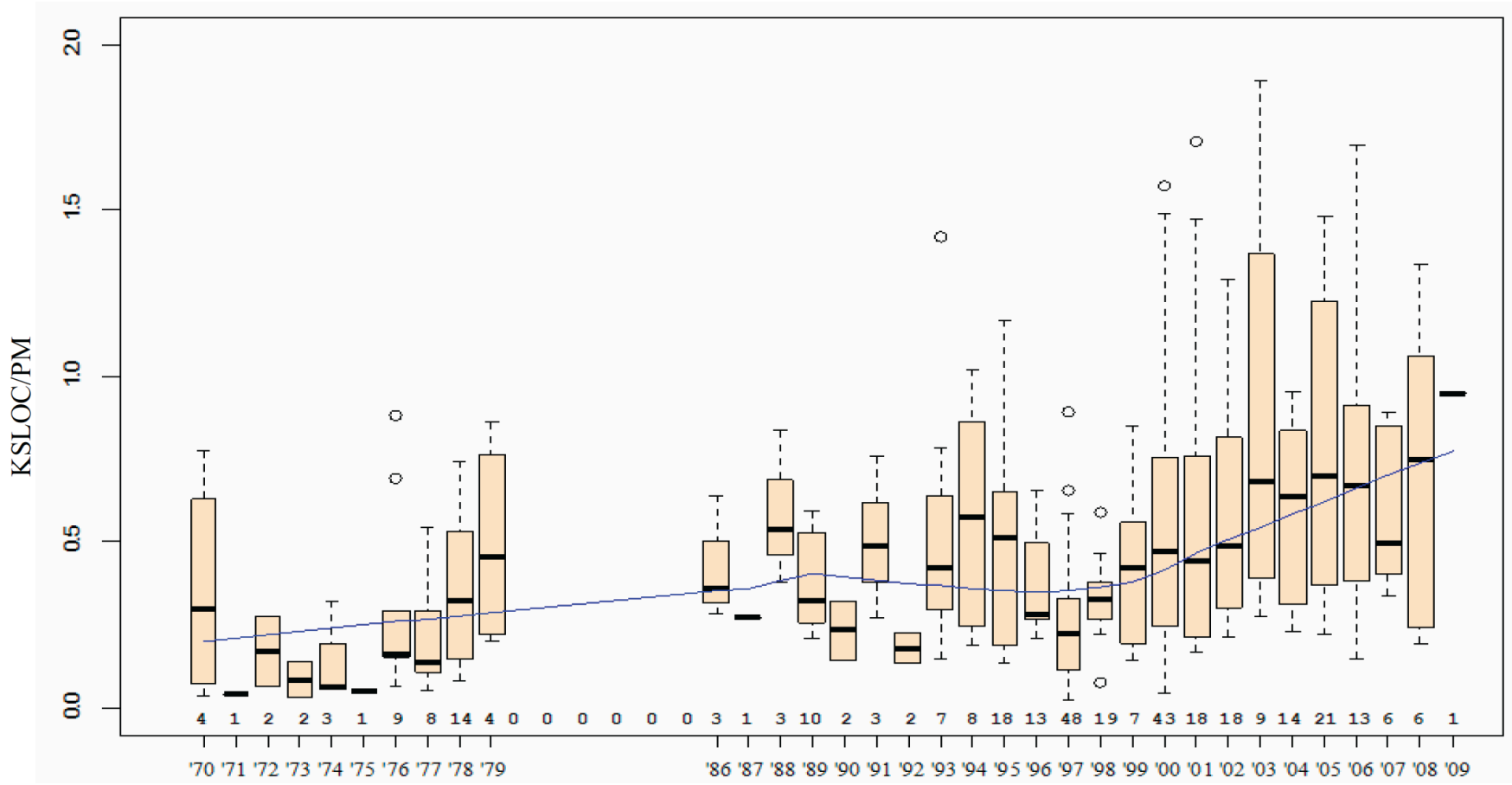
- Software is hard! And complex!
- Constantly changing technology
  ➔ no permanent platform to build on
- Software mostly done and taught the wrong way
- Most programmers are not good.
- Lack of first principles (laws)
- Lack of measurable standard unit of work
- Most don’t understand SW is art & engineering

and software reqs increasingly complex

- Real-time demands
- Cross-platform requirements
  → browsers, smart phones, tablets
- Concurrency and parallelism
- Internetworking / distributed systems
- Mobile, hand-held Systems
- Security

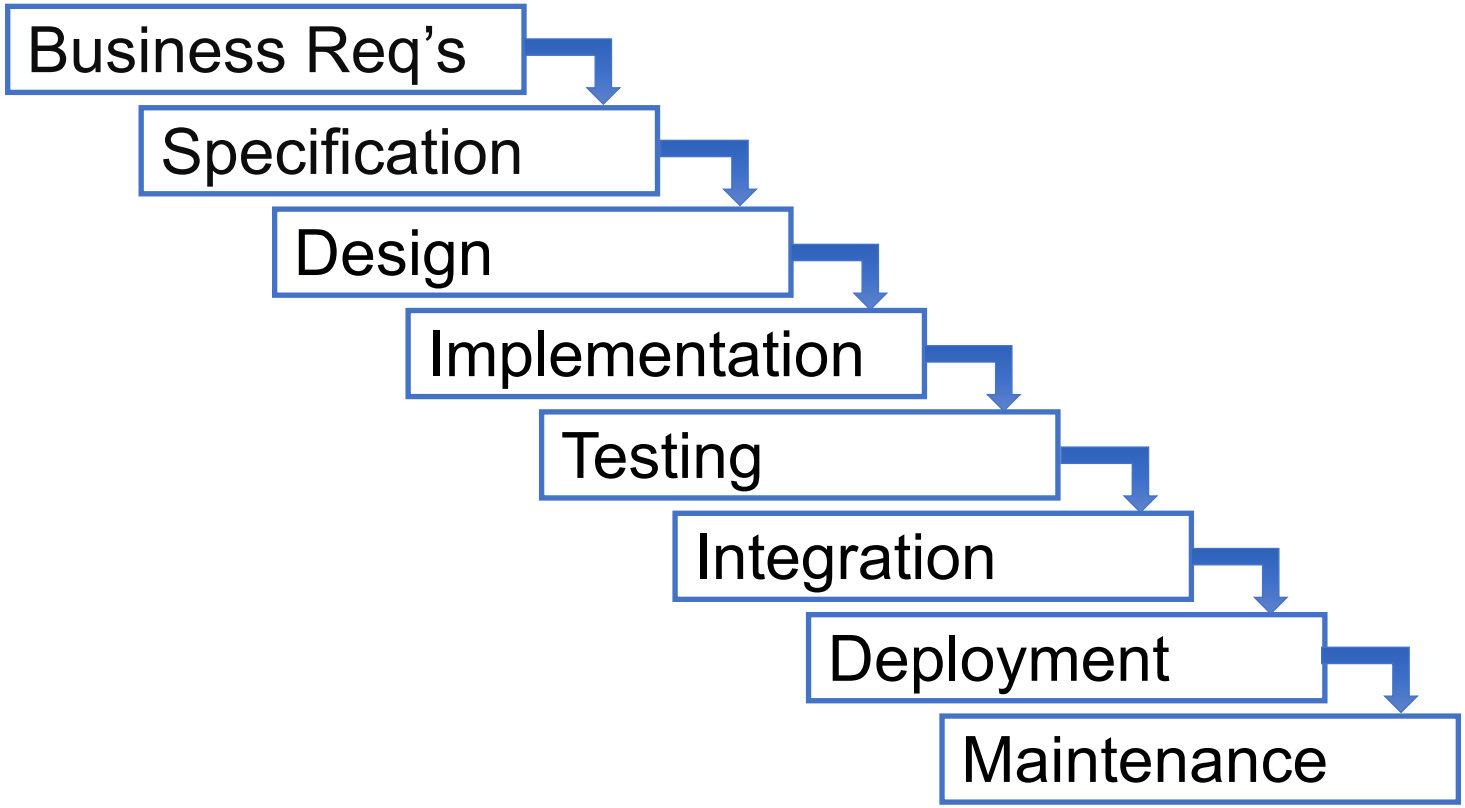
Software productivity mostly flat

… despite decades of research, new languages, processes, formal methods, dev tools, etc. etc.



Waterfall model

- Engineering approach: plan and document
- Approach followed by most (big) companies

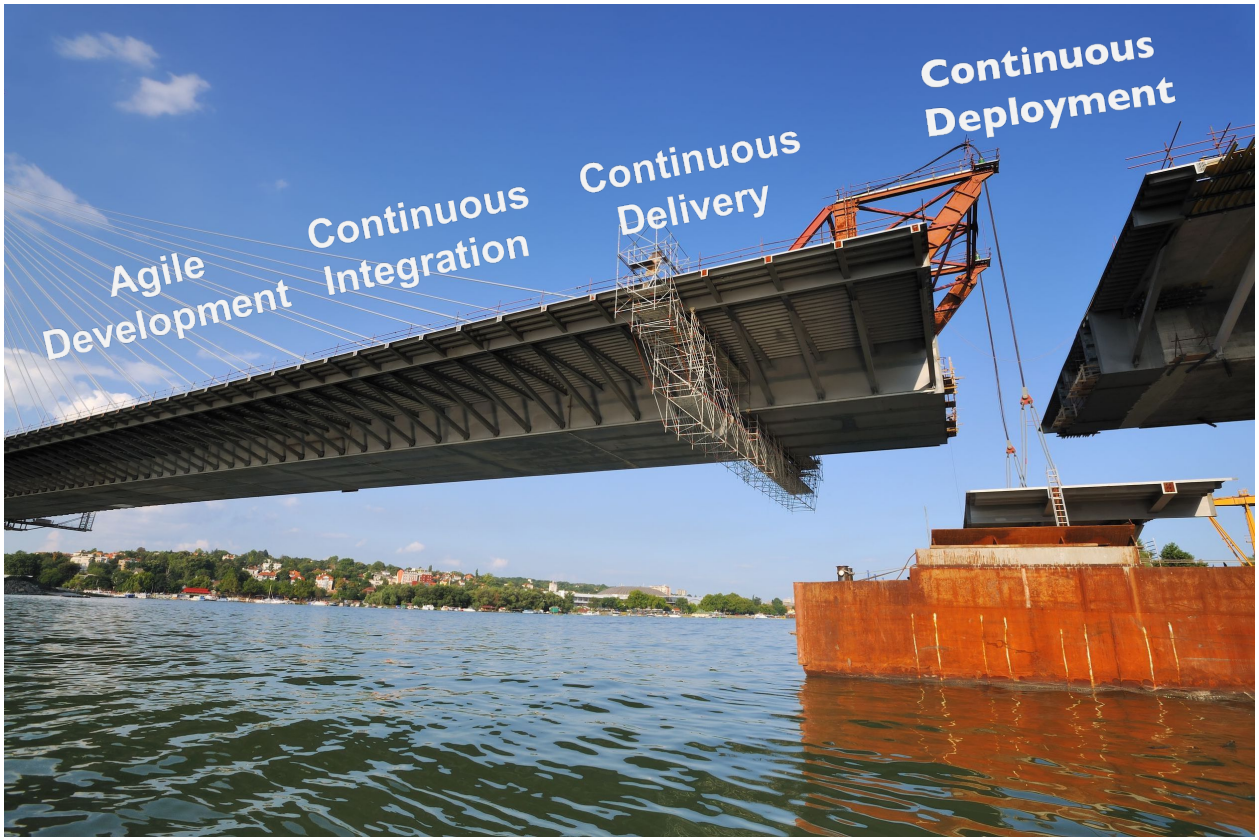


But…

- Properties of software vastly different than for traditional engineering systems
- Software systems are abstract and intangible; not constrained by physical laws or manufacturing processes
- No natural limits to the potential of software e.g., singularity
- It probably doesn't make sense to apply typical engineering methods and processes to software. But that is what tradition SE tells us to do!

We will teach a different approach

Agile Development
Continuous Integration
Continuous Delivery
Continuous Deployment



- self-organizing, cross-functional team
- practice of deploying small, incremental software updates to production
- within hours of creation
- by developers
- with no testing team

The different approach…

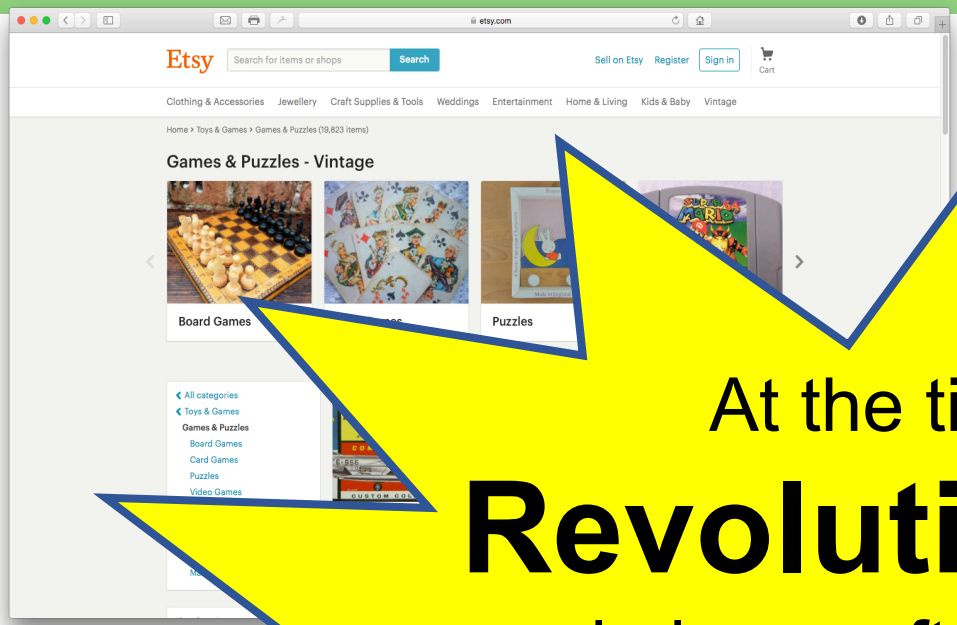
Embraced by more and more companies

facebook Etsy imvu wealthfront flickr

OANDA NETFLIX box TESLA hp

Google amazon airbnb GitLab

Example: first day working at Etsy.com



At the time

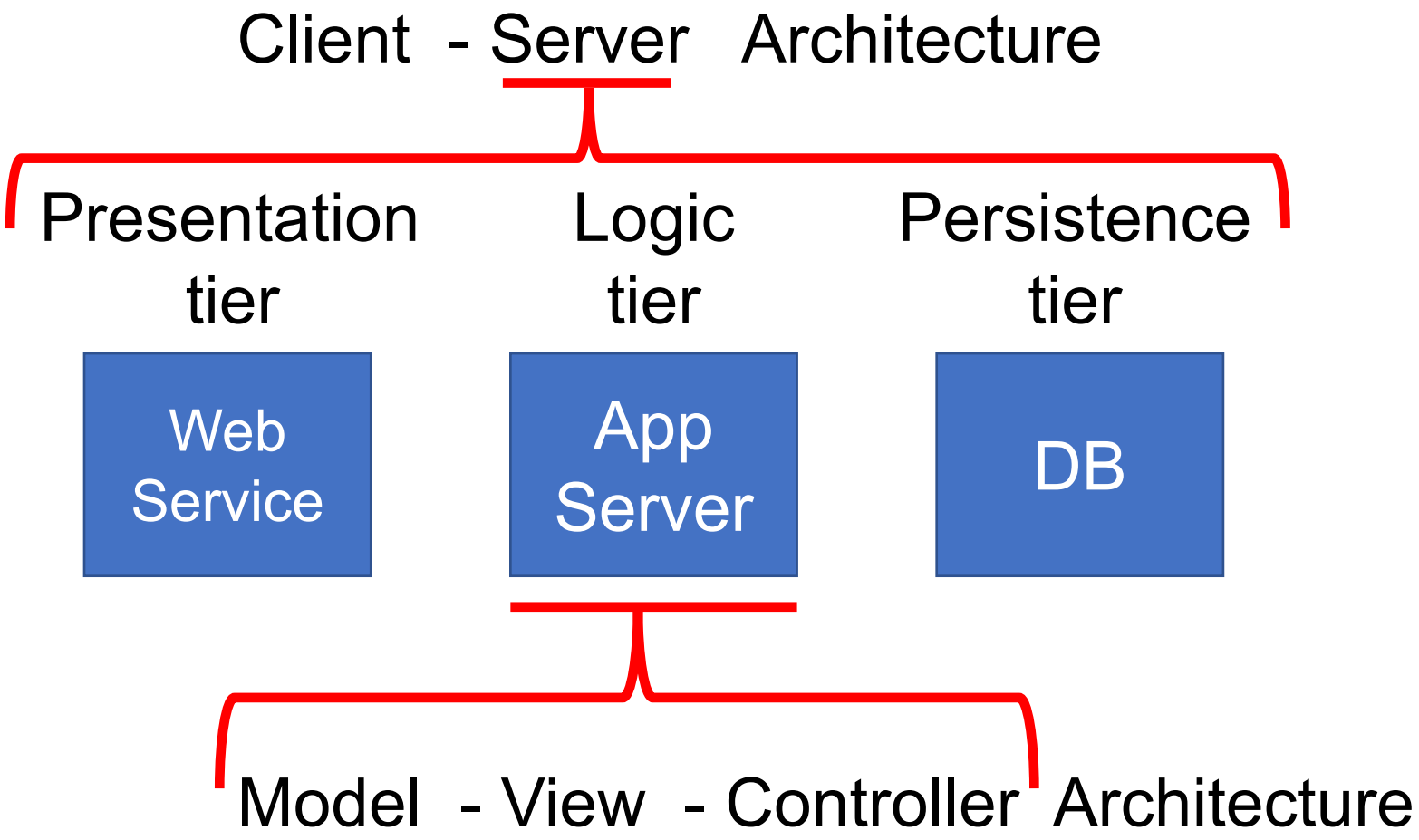
Revolutionary

in how software is produced and maintained

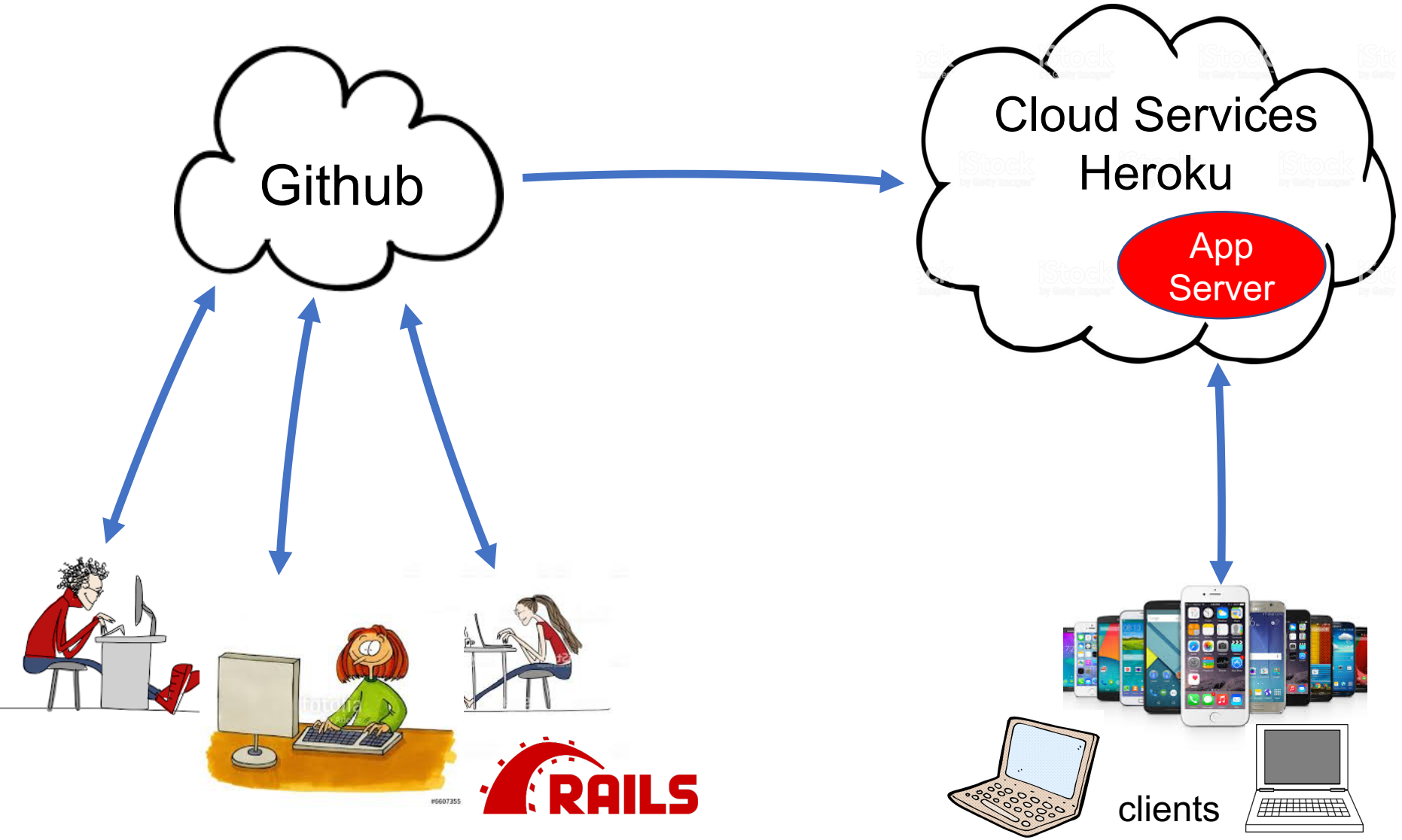
We will primarily learn by doing

Design and Implement Online Service-oriented App

Using:



Using Agile approach…

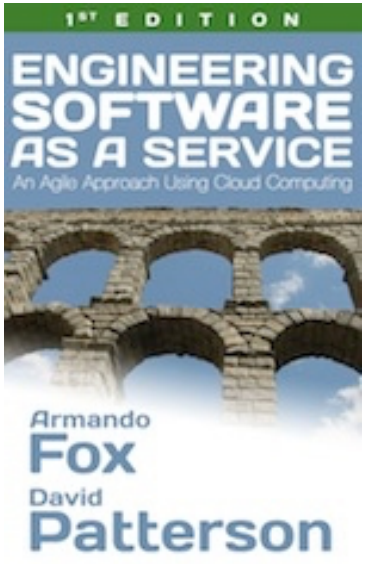


Homework

Come up with an online service worth having

Resources

- Ruby:
  - http://ruby-lang.org
  - http://ruby-doc.org
  - https://ruby.github.io/TryRuby/
  - https://rubygems.org - for gems
- Rails:
  - https://rubyonrails.org



Mark Composition

- Project & Homework: 45%
- Midterm & Tests: 20%
- Final Exam: 35%