SOFTWARE ENGINEERING

What is it, SE development lifecycles, and what to expect in the class

WELCOME!

- Meet me
  - UCCS 6.5 years
  - Prior: Undergrad: Allegheny College, Meadville, PA
    - Research fun!
  - Datatel, IU4, IT, etc…
    - Masters and PhD at University of Virginia
      - Research more fun! Computer Architecture and Software Testing/ Engineering emphasis
      - Software Testing for Mobile Devices and Embedded Systems research
- Meet your TA
  - Thomas Hastings from Mitre and current PhD student

> When I entered the world of software development, somehow I thought I would have to switch to an ergonomic keyboard to avoid some sort of occupational hazard of typing too much code everyday. The reality of software development: my day is spent making sure tests have been written and are passing on our projects, lawyers are cool with the libraries we chose to use for our project, the Scrum/Kanban board has been updated, discovering why the build is broken on our CI machine, and helping to debug my team members' code."

-*Dr. Robert Dickerson*

UVA PhD

Advisory Software Engineer at IBM

Former Lecturer (Computer Science) at The University of Texas at Austin

Former Visiting Assistant Professor (Computer Science) at William & Mary

WHAT IS SOFTWARE ENGINEERING?

SOFTWARE DEVELOPMENT LIFECYCLES

- Waterfall
  - Each task is completed 1 at a time
  - Each task takes 1-3 years
  - LOTS of documentation (Plan and Document organization)
- Spiral
  - Huh- customers want to see progress!
  - Each cycle (including all tasks) still takes 1-2 years
  - Added aspect of prototypes
  - Some or LOTS of documentation - different each iteration, but same amount of documentation in the end. (Plan and Document organization)
- Agile
  - 1-2 week iterations
  - In each iteration, plan, design, and implement 1 feature with working product at the end of each.

- In all: Requirements, Design (High level, then low), Implementation, Testing, Maintenance

CHALLENGES- TO NAME A FEW

- Your customer
  - Who are your stakeholders?
  - How do you find out what they actually want?
  - How do you prioritize their wants?
  - How do you stay in contact and get feedback?
  - What if they want changes? How can you adapt and keep the product in control?

CHALLENGES- TO NAME A FEW

- Design
  - High level
    - What model best fits the project's needs?
    - What tools support creation of that model?
    - How do you pick?
  - Low level
    - What models do you need for the project? (e.g. Object-oriented? Procedural?)
    - What language(s) support your needs?
    - How do you pick and start "outlining" your code?

CHALLENGES- TO NAME A FEW

- Implementation
  - Language selection
  - Architecture selection
  - Code, code, code…..
  - Debugging processes

CHALLENGES- TO NAME A FEW

- TESTING
  - No one likes to test…
  - Integration level testing
    - Easier and covers more code
    - Doesn't have much likelihood of hitting major bugs…
  - Unit level testing
    - Lot of thought needed!
    - A lot more challenging to write
    - In terms of metrics, slower progress
- Benefits: Self-documenting, outlines overall program behavior, helps others in future, etc…
- Benefits of Unit testing: Bugs less hidden, forces you to think through all scenarios, changes or additions that introduce bugs are FAR easier to see, and more…

CHALLENGES- TO NAME A FEW

- Maintenance
  - Less of a concern in today's environments
  - Maintenance often folded into the testing process
  - In waterfall models, still VERY important and involves much testing and **refactoring**

HOW AGILE HELPS

- Emphasizes short cycles (1-2 weeks)
- Individuals and Interactions over processes and tools
- Working Software over comprehensive documentation
- Customer Collaboration over contract negotiation
- Responding to Change over following a plan

VERTICAL OVER HORIZONTAL SLICES

- Plan, design, and implement feature(s)
- Do not design the whole product first
- Pros and Cons to this?

WHAT WE DO IN THIS CLASS

- Agile development methodologies
- Deploy to the cloud regularly
- Get customer feedback each iteration
- Learn to create “beautiful code”
- Code itself
- Tests - how and why?
- Documentation?
- Learn to design and refactor code to make it more beautiful
- Learn to work in teams and adapt QUICKLY

WHAT WE DO IN THIS CLASS... CNTD

- BIG class projects
  - Prepare and understand the main tools before starting the project (2 HW assignments)
  - Work in 2 week iterations
  - One-pizza teams of 5-6
  - Frequent team and customer feedback
  - Demos each iteration with industry collaborators present!
  - Plan, implement, thoroughly test, deploy each iteration using clear project management

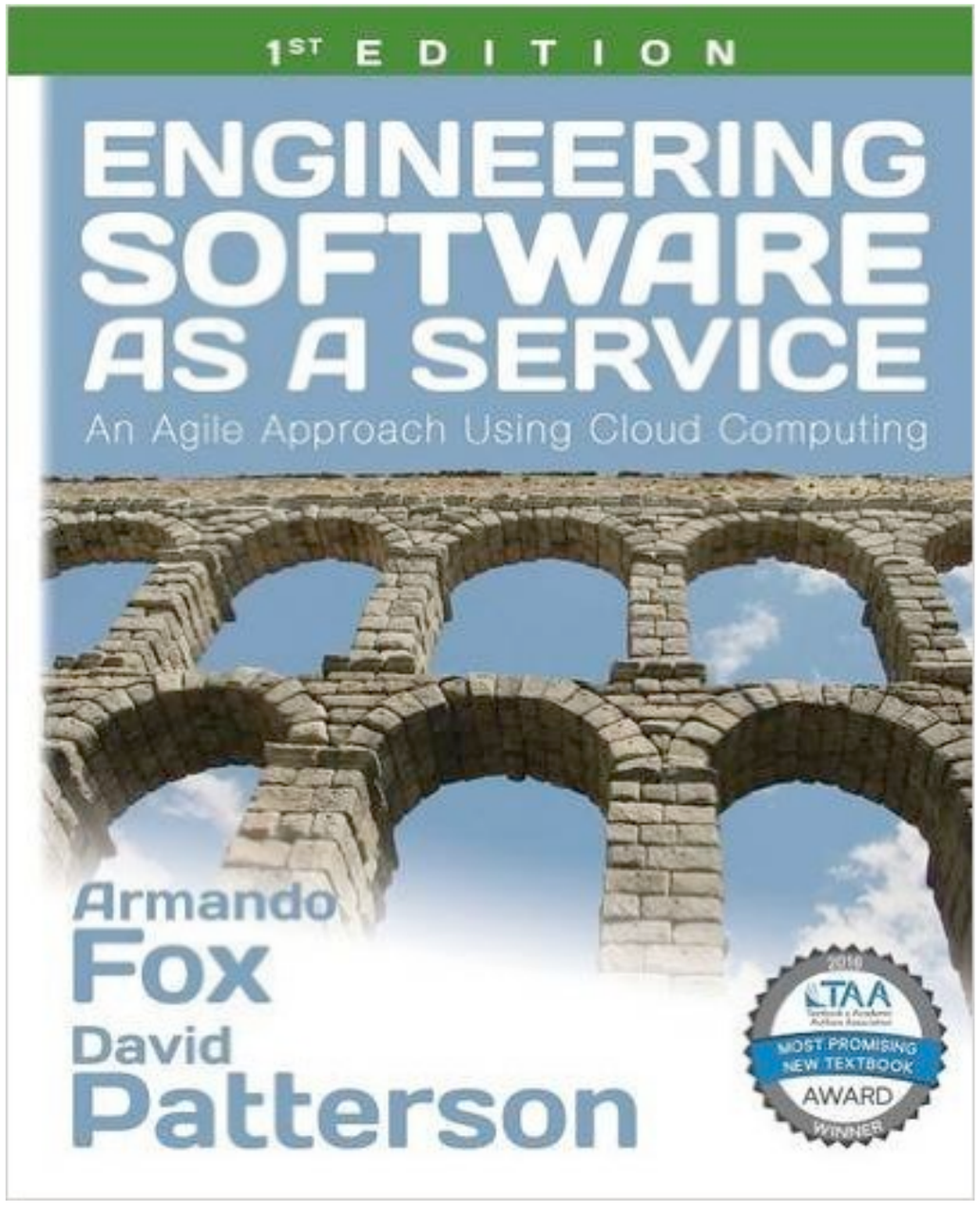
THINGS WE'LL AVOID...

What we say	What we mean
Horrible hack	Horrible hack that I didn't write
Temporary workaround	Horrible hack that I wrote
It's broken	There are bugs in your code
It has a few issues	There are bugs in my code
Obscure	Someone else's code doesn't have comments
Self-documenting	My code doesn't have comments
That's why it's an awesome language	It's my favorite language and it's really easy to do something in it.
You're thinking in the wrong mindset	It's my favorite language and it's really hard to do something in it.
I can read this Perl script	I wrote this Perl script
I can't read this Perl script	I didn't write this Perl script
Bad structure	Someone else's code is badly organized
Complex structure	My code is badly organized
Bug	The absence of a feature I like
Out of scope	The absence of a feature I don't like
Clean solution	It works and I understand it
We need to rewrite it	It works but I don't understand it
emacs is better than vi	It's too peaceful here, let's start a flame war
vi is better than emacs	It's too peaceful here, let's start a flame war
IMHO	You are wrong
Legacy code	It works, but no one knows how
^X^Cquit^\[ESC][ESC]^C	I don't know how to quit vi

ASSHOLE DRIVEN DEVELOPMENT– WHAT OFTEN HAPPENS

- From http://scottberkun.com/2007/asshole-driven-development/
- **Asshole-Driven development (ADD)** – Any team where the biggest jerk makes all the big decisions is asshole driven development. All wisdom, logic or process goes out the window when Mr. Asshole is in the room, doing whatever idiotic, selfish thing he thinks is best. There may rules and processes, but Mr. A breaks them and people follow anyway.
- **Cognitive Dissonance development (CDD)** – In any organization where there are two or more divergent beliefs on how software should be made. The tension between those beliefs, as it's fought out in various meetings and individual decisions by players on both sides, defines the project more than any individual belief itself.
- **Cover Your Ass Engineering (CYAE)** – The driving force behind most individual efforts is to make sure than when the shit hits the fan, they are not to blame.
- **Development By Denial (DBD)** – Everybody pretends there is a method for what's being done, and that things are going ok, when in reality, things are a mess and the process is on the floor. The worse things get, the more people depend on their denial of what's really happening, or their isolation in their own small part of the project, to survive.
- **Get Me Promoted Methodology (GMPM)** – People write code and design things to increase their visibility, satisfy their boss's whims, and accelerate their path to a raise or the corner office no matter how far outside of stated goals their efforts go. This includes allowing disasters to happen so people can be heroes, writing hacks that look great in the short term but crumble after the individual has moved on, and focusing more on the surface of work than its value.

BOOK AND HOW IT'LL BE USED



- Get the FIRST edition
- Work through the examples and videos provided
  - HUGE network of support!
- Electronic version encouraged!
- Check out the MOOC versions too!

EXPECTATIONS AND WORK

- In-class activities and participation: 2% (includes surveys!)
- Homeworks (2): 18%
- Tests (2): 28%
- Project (in teams): 50%
  - Includes presentations, write-ups, iteration submissions (individual student evaluation based on overall group evaluation and individual effort: see below)
- Individual Write-up: 2%

EXPECTATIONS AND WORK… CNTD

- Deadlines
  - See the Calendar in Canvas
  - Read the Syllabus for important info on responsibilities!!
    - Note the bold sections!!
- Paired Programming- use it!

PROJECT

- Teams have been selected
  - See Canvas “Project” for details
  - Groups will soon be created in Piazza
  - Groups will soon be added in Github so you can work together
- Starting the project next week! Start thinking of neat things you’d want others to develop
  - Example past projects (random sampling!):
    - Beer Me
    - Zombie Apocalypse
    - Hiker Hero
    - Food Pairing
    - Fridge Friend
    - Medication Warning Organizer
    - UCCS Bus Schedules

PROJECT

- Demos
  - At the end of each iteration, industry representative will attend class to see your demos and give a brief talk
  - Industry representatives include a speakers from Polaris, Lockheed Martin, Harris, etc…
- Make your project great at the end of each iteration- function, design, well-tested!

TOOLS- REQUIRED

- Ruby (Language)
- Rails (Architectural framework)
- Class-provided VM (Development/Deployment infrastructure)
- Cucumber (Integration-level testing tool)
- RSpec (Integration and unit-level testing tool)
- Github (Version control tool)
- ERB/HTML/Javascript/JQuery/XML/AJAX/others….
- Asana/Zenhub (Project management tool)
- Other:
  - Teammates (Project/Individual evaluation)
  - Piazza (Class discussion)
- Perhaps others based on your projects!
  - Frequently used: Selenium, Wireshark, Gumby, Devise, Turbolinks, Textacular, Guard, etc….
  - Plus APIs of your choosing!

USING PIAZZA

- USE IT!
- Ask questions
- Answer questions!
- Post cool stuff/tutorials/videos
- Notes:
  - Posts can be anonymous
  - Posts can be sent to just the instructor/TAs/helpers
    - If you want it to go just to me, email me with CS3300 in your subject line- I prefer that you use Piazza for anything not personal.
  - Posts can be sent to groups (Developers, Customers&Developers)
- Anything with crude or insulting language will be moved to private or deleted- stay professional!