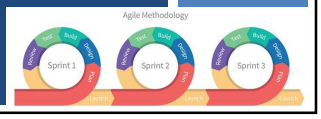
TCSS 360: SOFTWARE DEVELOPMENT AND QUALITY ASSURANCE

Introduction to Software Engineering

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma



TCSS 360 C – WINTER 2017

- Introduce practices of Software Engineering
- In the context for developing applications with service oriented architectures (SOA)
- Much service oriented software is hosted in the cloud
- Not a programming course per se
  - Examples in Java/Javascript, Ruby/Rails
- Group projects to practice an Agile process to develop service oriented software

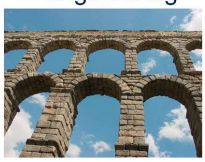
MISSING SOFTWARE SKILLS

- Working with legacy code, refactoring
- Working with non-technical customers
  - Years spent learning programming, algorithms, computer science theory
  - How do we put this knowledge to work?
- Bridging the gap between theory and practice
- How to test software

TEXTBOOK: WWW.SAASBOOK.INFO

- $10 Kindle
- $39.99 print
- $30 used
- Amazon
- 4.4 / 5 stars

Introduction to Software Engineering



(*Engineering Software as a Service* § 1.2.1)

Armando Fox and David Patterson

ONLINE RESOURCES

- TCSS360
- http://www.sasbook.info/students
- Online "get started" tutorials on ruby in case want to get feet wet before Ruby lectures
- Tutorials on other tools (GitHub, etc.)

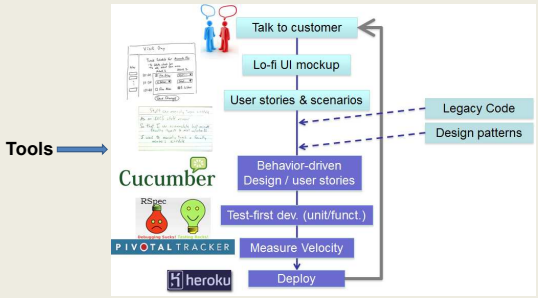
OBJECTIVES

- Introduction, Syllabus
- Software Development Processes
  - Plan and Document
  - Agile
- Service Oriented Architecture
- Cloud Computing
- Software Quality Assurance - Testing

AGILE SOFTWARE DEVELOPMENT

© Scott Adams, Inc./Dist. by UFS, Inc.

AN AGILE SOFTWARE PROCESS



WHY SOFTWARE ENGINEERING?

- Margaret Hamilton, 1965
- Director & Supervisor of Software Programming for Project Apollo
- Worked on priority-based asynchronous scheduling software
- Prevented last-minute abort of the first moon landing
- Coined term "Software Engineering", which gave name to first-ever conference on topic (1968)
- …convened to address the "software crisis"

RANKING TOP 200 JOBS (2012)

1. Software Engineer
28. Civil Engineer
34. Programmer
40. Physician
47. Accountant
60. Mechanic
73. Electrical
87. Attorney

Researches, designs, develops and maintains software systems along with hardware development for medical, scientific, and industrial purposes. Income: $88,142 (+25%)

Organizes and lists the instructions for computers to process data and solve problems in logical order. Income: $71,178

…Flight Attendant
…er
…er Reporter
200. Lumberjack

InformationWeek 5/15/12. Based on salary, stress levels, hiring outlook, physical demands, and work environment (www.careercast.com)

HEALTHCARE.GOV

HEALTHCARE.GOV - 2

- "The DOD is hampered by a culture of acquisition-related practices that favor large programs, high-level oversight, and a very deliberate, serial approach to development and testing. (the waterfall model)

  Programs (contractors) are expected to deliver complete, nearly perfect solutions that take years to develop are the norm in the DOD.

  These approaches run counter to Agile practices in which the product is the primary focus, where end users are engaged early and often – the oversight of incremental product development is delegated to the lowest practical level…"

  (National Research Council 2010)

HOW TO AVOID INFAMY?

- Consider lessons of 60 years of SW development
- Software engineering is more than just programming
- Consider many variations of software processes
- Make software development as predictable as building a bridge
- What development processes make building software more predictable?
  - Plan and Document, Agile methods

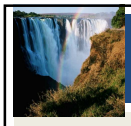
IEEE SPECTRUM: SOFTWARE WALL OF SHAME

YEAR	COMPANY	OUTCOME (COSTS IN US $)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to $33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to $3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after $54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately $400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing $527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to $160 million loss.
2003–04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of $100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after $170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after $33.2 million† is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to $445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to $100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after $130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing $25 million.
1999	United Way	Administrative processing system canceled after $12 million is spent.
1999	State of Mississippi	Tax system canceled after $11.2 million is spent; state receives $185 million damages.

PLAN AND DOCUMENT PROCESSES

SOFTWARE ENGINEERING: PLAN AND DOCUMENT PROCESSES

- Bring engineering discipline to SW
  - Term coined ~ 20 years after 1st computer
  - Find SW development methods as predictable in quality, cost, and time as civil engineering
- "Plan-and-Document"
  - Before coding, project manager makes plan
  - Write detailed documentation all phases of plan
  - Progress measured against the plan
  - Changes to project must be reflected in documentation and possibly to plan



PLAN AND DOCUMENT: WATERFALL VS. SPIRAL

Common objective: the earlier an error is found the cheaper it is to fix

Waterfall

- Sequence of phases 6 to 18 month total length
- Best when requirements are fixed, unchanging

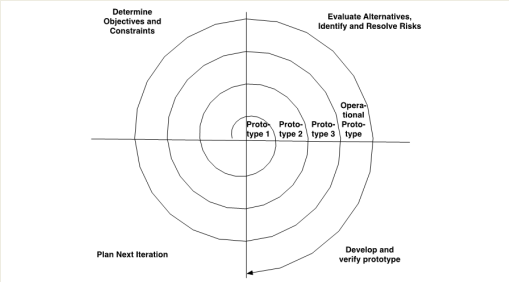
1. Requirements analysis
2. Architectural design
3. Implementation, integration
4. Verification
5. Operation, Maintenance

Spiral

- Iterations of 6-24 months
- Each builds prototype
  - Captures requirements

1. Determine objectives and constraints
2. Evaluate alternatives, identify risks
3. Develop and verify prototype
4. Plan the next iteration

SPIRAL LIFECYCLE



WHAT PHASE OF SOFTWARE DEVELOPMENT CONSUMES THE MOST RESOURCES?

- ❑ Design
- ❑ Development
- ❑ Testing
- ❑ Maintenance
- 60% of costs are maintenance
- Many first software development jobs involve maintenance, feature enhancement, bug fixing
- Most of a CS curriculums avoids software maintenance, refactoring

ROLE OF PROJECT MANAGERS

- Plan and Document software process depends on top-notch project managers
- Key tasks
  - Write contract to win the project
  - Recruit development team
  - Evaluate software engineers performance, set salary
  - Estimate costs, maintain schedule, manage budget, evaluate risks & overcomes them
  - Document project management plan
  - Gets credit for success, or blamed if projects are late or over budget

HOW WELL DOES WATERFALL WORK?

- And the users exclaimed with a laugh and a taunt: "It's just what we asked for, but not what we want." —Anonymous
- "Plan to throw one [implementation] away; you will anyhow."
- Fred Brooks, Jr. Author: Mythical Man Month
- No Silver Bullet - essay (1999 Turing Award winner)
- Often after build first one, developers learn right way they should have built it

(Photo by Carola Lauber of SD&M www.sdm.de. Used by permission under CC-BY-SA-3.0.)

FRED BROOKS

"Adding manpower to a late software project makes it later."
Fred Brooks, Jr.,
The Mythical Man-Month

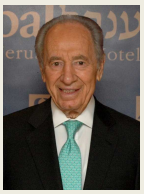
- It takes time for new people to learn project, domain
- Communication overhead grows with team size
  - Leaves less time for work
- Ideal may be groups of 4 to 9 people
  - Hierarchically composed for larger projects

PLAN AND DOCUMENT PITFALLS

- Can plan & development practices deliver on cost, schedule, and quality targets?
- Required
  - Extensive documentation
  - Planning
  - Experienced project managers
- Consider
  - Ratio of documentation to code with plan and document approaches
  - How can documentation be reduced while eliciting, capturing, and delivering on customer software requirements?
  - How to avoid hacking?

PERES'S LAW

"If a problem has no solution,
it may not be a problem,
but a fact, not to be solved,
but to be coped with over time."
— Shimon Peres
(winner of 1994 Nobel Peace Prize for Oslo accords)



(Photo Source: Michael Thaidigsmann, put in public domain, See http://en.wikipedia.org/wiki/File:Shimon_peres_wjc_90126.jpg)

AGILE SOFTWARE PROCESS

L1.25

AGILE MANIFESTO - 2001

"We are uncovering better ways of developing SW by doing it and helping others do it. Through this work we have come to value

- **Individuals and interactions** over processes & tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

- While there is value in the items on the right we value the items on the left more.

EXTREME PROGRAMMING (XP) VARIANT OF AGILE LIFECYCLE

- If **short iterations** are good, make them as short as possible (weeks vs. years)
- If **simplicity** is good, always do the simplest thing that could possibly work
- If **testing** is good, test all the time. Write the test code before you write the code to test.
- If **code reviews** are good, review code continuously, by programming in pairs, taking turns looking over each other's shoulders.
- But you have to do all of them.

AGILE LIFECYCLE

- Embraces change as a fact of life: continuous improvement vs. phases
- Developers continuously refine working but incomplete prototype until customers are happy, with customer feedback on each **Iteration** (every ~1 to 2 weeks)

- Agile emphasizes **Test-Driven Development** (**TDD**) to reduce mistakes, **User Stories** to validate customer requirements, **Velocity** to measure progress

AGILE: THEN AND NOW

- Controversial when introduced, 2001
  - "… yet another attempt to undermine the discipline of software engineering… nothing more than an attempt to legitimize hacker behavior."
    - Steven Ratkin, "Manifesto Elicits Cynicism," *IEEE Computer*, 2001
- Mainstream more recently
  - 2012 study of 66 projects found majority using Agile, even for distributed teams

AGILE DRAWBACKS

- Fallacy: The Agile lifecycle is best for all software development
- Good match for some SW, especially SaaS
- But not for NASA, code subject to regulations
- In TCSS 360, we will practice Agile, but will also consider Plan & Document perspectives
- Software lifecycles (processes) are constantly evolving
- Expect to see new processes
- Many companies use variants or hybrids combining various aspects as best fit for project demands

WHICH SOFTWARE PROCESS?

Yes=Plan and document; No = Agile

1. Is specification required?
2. Are customers unavailable?
3. Is the system to be built large?
4. Is the system to be built complex (e.g., real time)?
5. Will it have a long product lifetime?
6. Are you using poor software tools?
7. Is the project team geographically distributed?
8. Is team part of a documentation-oriented culture?
9. Does the team have poor programming skills?
10. Is the system to be built subject to regulation?

WHAT'S MISSING FROM AGILE

- Requirements elicitation
- Documentation
- Progress estimation
- Unit & functional testing
- System / integration testing
- User acceptance testing
- Continuous refactoring of design

SOFTWARE-AS-A-SERVICE (SAAS) & SERVICE ORIENTED ARCHITECTURES (SOA)

WHY SAAS IS > SHRINK-WRAPPED SW…

1. No install worries about HW capability, OS
2. Data stored safely, persistently on servers
3. Easy for groups to interact with same data
4. If data is large or changed frequently, simpler to keep 1 copy at central site
5. 1 copy of SW, single HW/OS environment => no compatibility hassles for developers
   => beta test new features on 1% of users
6. 1 copy => simplifies upgrades for developers *and* no user upgrade requests

SHRINK WRAPPED SOFTWARE (SWS)

- Client-specific binary, frequent upgrades
- Must work w/many versions of HW, OS, Libraries…
- Hard to maintain
- Extensive compatibility testing per release
- Alternative: server-centric app, thin client
- Search, email, commerce, social nets, video…
- Now also productivity (Google Docs/Office 365), finance (TurboTax Online), IDEs (Codenvy)…

BUILDING LARGE SAAS

- Can you design software so that you can recombine independent modules to offer many different apps without a lot of programming?
  - Solves "Agile only good for small teams"
- "[Amazon CEO Jeff Bezos] realized long before the vast majority of Amazonians that Amazon needs to be a platform."

Steve Yegge, Googler, former Amazonian, in a 2011 blog post

2002 JEFF BEZOZ EMAIL: AMAZON SERVICES MANDATE

1. "All teams will henceforth expose their data and functionality through service interfaces."
2. "Teams must communicate with each other through these interfaces."
3. "There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network."

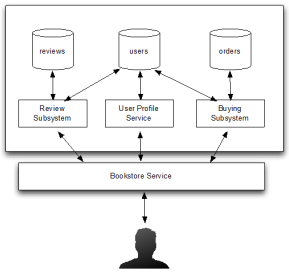
AMAZON SW MANDATE – 2

4. "It doesn't matter what [API protocol] technology you use."
5. "Service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions."
6. "Anyone who doesn't do this will be fired."
7. "Thank you; have a nice day!"

BOOKSTORE – SILO DESIGN

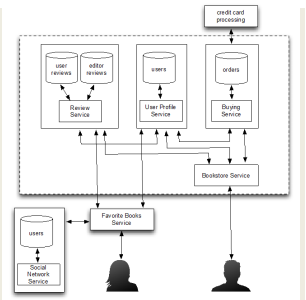
- Internal subsystems share data directly
  - DBs shared internally
- For example: Each subsystem accesses the users DB
- All subsystems composed together in a single API

("Bookstore")



BOOKSTORE – SOA DESIGN

- Subsystems are independent, as if in separate datacenters
- Review Service access User Service API
- Can recombine to make new service ("Favorite Books")



*WHICH OF THE FOLLOWING IS A **DISADVANTAGE** OF SOA COMPARED TO A SILO DESIGN?*

- ❑ SOA may be harder to debug and tune
- ❑ SOA results in lower developer productivity
- ❑ SOA's complexity is a poor match for small teams
- ❑ SOA is more expensive to deploy than SILO as more servers are needed to handle the same workload
- SOA is harder to debug and tune: SOA can experience partial failures as a complete system consists of a number of microservices hosted separately, but composed together

QUESTIONS