SOFTWARE ENGINEERING (CS3053)

Introduction

Fall 2016

Software Engineers

- A Software Engineer is a developer which utilizes engineering tools and techniques in order to develop software
- Some "definitions" from around the web:
  - Dijkstra: ’If you carefully read its literature and analyze what its devotees actually do, you will discover that software engineering has accepted as its charter, “How to program if you cannot.”’
  - On Stack Exchange: "Software engineers" are engineers in the same way that "sandwich artists" are artists. It’s marketing and/or ego applied to job titles

Examples of Failed Software Engineering

- 1985 - Therac-25 lethal radiation overdose
  - Concurrent programming errors - 3 died
- 1999 - Mars Climate Orbiter disintegration
  - Wrong measuring units - lost of $325M
- 2005 - FBI Virtual Case File project abandonment
  - Gave up after 5 years of work - lost of $170M
- 1996 - Ariane 5 rocket explosion
  - Conversion from 64 bits to 16 bits numbers - lost of $500M

Is it so bad?

- Many failed software engineering projects costing tens of billions of dollars
- Are Dijkstra and the others right?
- Can't we make building software as accurate and predictable as building a bridge?
- If so, what development process do we need to use?

In the 70s

- Software got more and more complex
- Contractors should estimate time and costs
- Adapt to software development methods from civil engineering
- Plan and Document
  - Start by making a plan and estimate costs
  - Document precisely every detail in the plan
  - Measure progress against the plan
  - Any change must be reflected in plan and documentation

The "Waterfall" Approach (1970)

- One release
  1. Requirements analysis and specification
  2. Architectural design
  3. Implementation and Integration
  4. Verification
  5. Operation and Maintenance
- Sequential - Earlier is cheaper
- Life span: years
- Does it work?
  - Customer might change their mind
  - “Plan to throw one [implementation] away; you will, anyhow." - Fred Brooks, Jr. (1999 Turing Award winner)

The "Spiral" Approach (1986)

- Several iterations
  1. Determine objectives and constraints
  2. Evaluate alternatives and identify risks
  3. Develop and verify the prototype for this iteration
  4. Plan next iteration

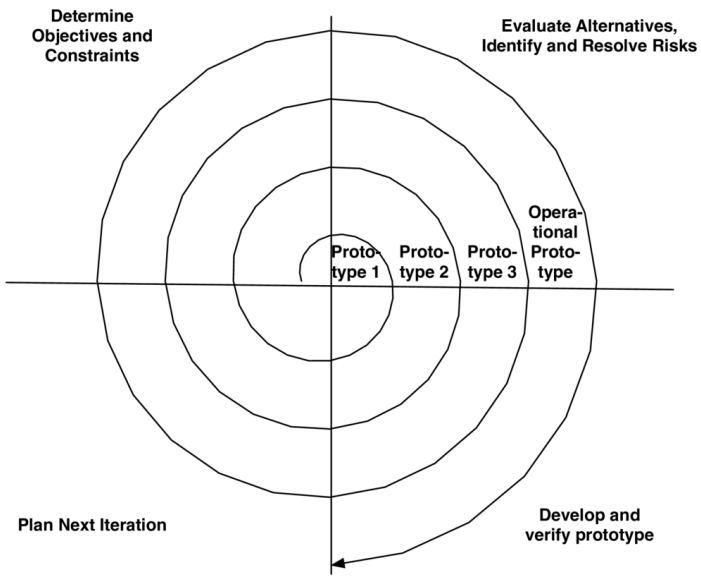

Figure 1.2 from SaaS book

The "Spiral" Approach - Good and Bad

- The good
  - Time for customers to change their mind
  - Estimations becomes more realistic after each iteration
  - Knowledge accumulated
  - Higher change of success
- The bad
  - Iterations are still heavy
    - 6 to 24 months long
    - Lots of documentation per iteration
    - Higher total cost

Properties of Plan and Document Projects

- Very hierarchical
- Lots of documentation
- High manpower
- Members change during project
- Separate roles
- Infrequent contact with client

Success Rate of Plan and Document Projects

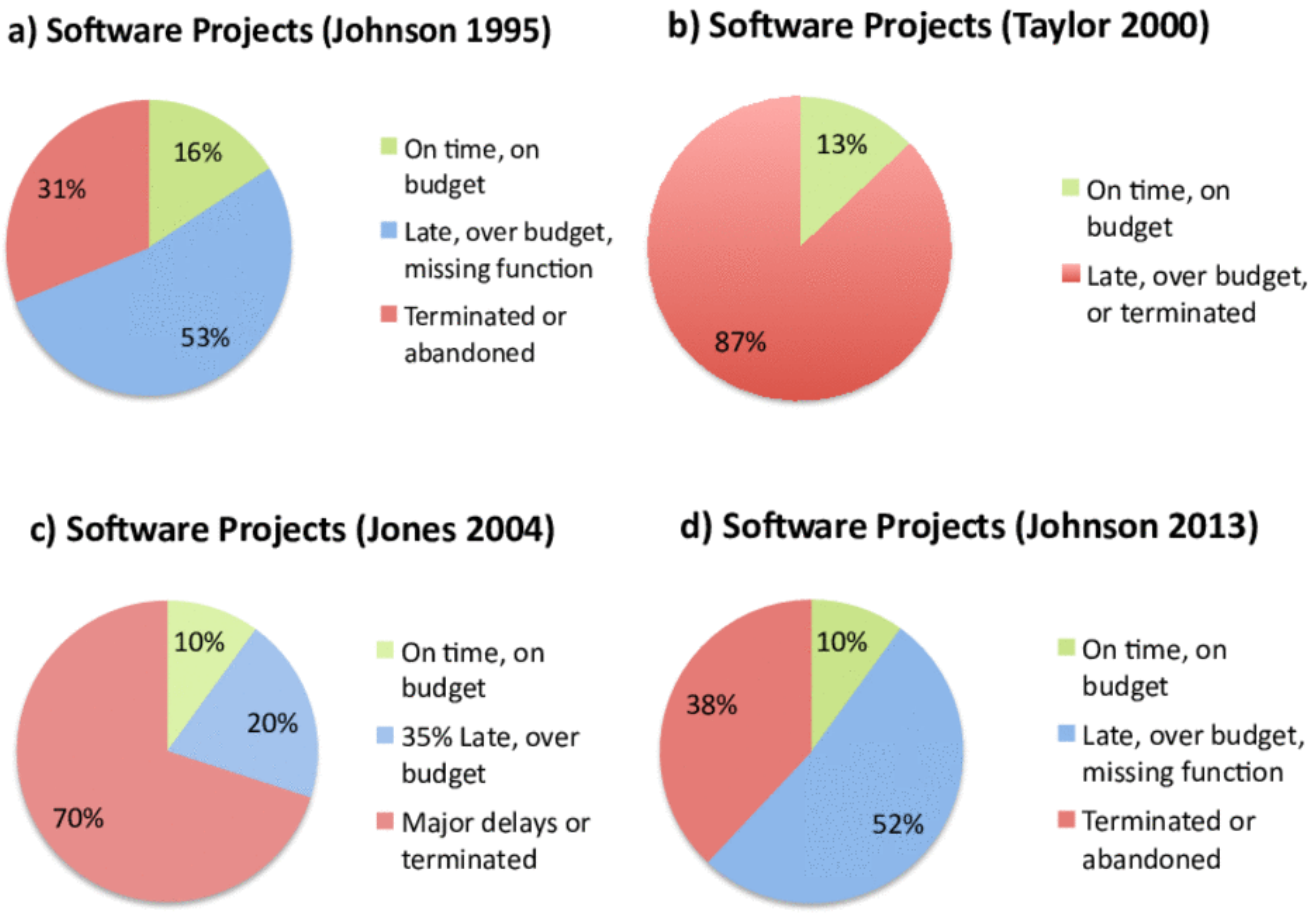

Figure 1.4 from SaaS book

What can we do?

- We tried to optimize Plan and Document but still low success rate
- We want to reduce documentation
- We don’t want to depend on a very good project manager
- But without careful planning, documentation and a good manager, would it be just hacking?

The "Agile" Manifesto

- What can we change?
- Agile Manifesto (2001):
  - Individuals and interaction vs Processes and tools
  - Working software vs Comprehensive Documentation
  - Customer collaboration vs Contract negotiation
  - Respond to change vs Follow a plan

The "Agile" Approach

- Continuous iterations
  - Collaboration with client
  - Develop and verify (incomplete) prototype
- Life span: few weeks
- Continuous improvements
- Smaller projects - better methodologies

Extreme Agile Programming (XP)

- If short iterations are good, make them even shorter (1-2 weeks)
- If simplicity is good, always choose the simplest solution
- If testing is good, test all the time
- If code reviews are good, review all the time

Agile Now and Then

- Steven Ratkin (2001):
  - "... yet another attempt to undermine the discipline of software engineering... nothing more than an attempt to legitimize hacker behavior."
- 2012 - A study of 66 projects found a majority using Agile
- A survey of 50,000 projects of under $1M budget

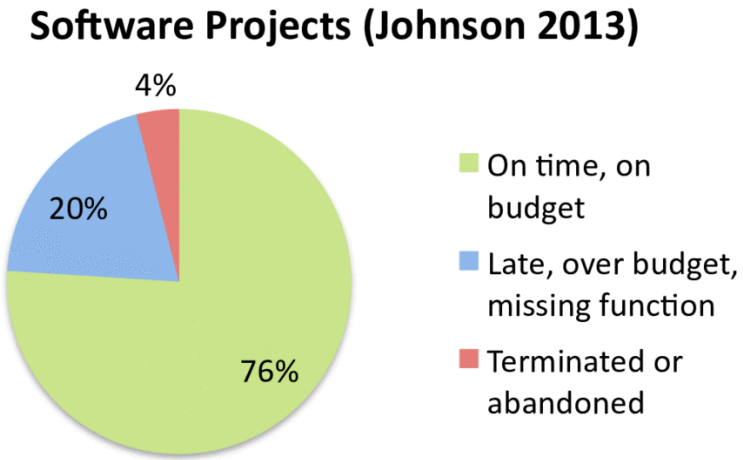

Figure 1.6 from SaaS book

Plan and Document or Agile?

Sommerville (2010)	
	Question (No suggests Agile, Yes suggests P&D)
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?

Table 1.5 from SaaS book

Administration

- Lecturer: Tomer Libal (tlibal@aup.edu)
- Grade: 10% midterm, 40% project, 30% HW and 20% participation in class
- Book: “Engineering Software as a Service: An Agile Approach Using Cloud Computing” by Armando Fox and David Patterson (versions 1.1 or 1.2)
- Additional resources: http://www.saasbook.info/students

Exercise 1

- Follow the instructions in the webpage
  - https://github.com/AUP-SE/intro