SOFTWARE ENGINEERING (CS3053)

SaaS: Software as a Service

Fall 2016

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

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

- Recent surveys show that Agile is being used in 60-80% of all programming teams!

Traditional Software

- Traditonal Software is given as a binary which is installed on client computer
  - Must work on many different operating systems and computers
  - Users are responsible for upgrade
  - New versions require extensive release cycle for ensuring compatibility
  - Customer support is difficult
  - Installation might be non trivial
- Is there an alternative?

Software as a Service

- SaaS: Software is a service accessed over internet via a thin program (i.e. browser) running on client device
  - Examples?
- SaaS also has versions of tradiotional software
  - Examples?
- Seems to be the future

Why SaaS?

1. No user installation
2. Central data storage
3. Cooperation
4. Data can be large and change frequently
5. Easier to market and demonstrate
6. Single copy of software
7. Pay per use
8. Harder to hack

Why not SaaS?

1. Requires Internet connection
2. Communication
3. Higher learning curve
4. More code

More Code..

- What can we do for large projects?
- Answer: Base it on many small independent programs
- How: Each program will (only) expose its functionality through a well defined API

Letter of Jeff Bezos - CEO of Amazon (2002)

1. All teams will henceforth expose their data and functionality through service interfaces
2. Teams must communicate with each other through these interfaces
3. . . . The only communication allowed is via service interface calls over the network
4. It doesn't matter what technology you use
5. Service interfaces, without exception, must be designed from the ground up to be externalizable . . .
6. Anyone who doesn't do this will be fired
7. Thank you; have a nice day!

Service Oriented Architecture (SOA)

- SOA: Software Architecture made of small modules designed to be services which communicate with each other through an API
- Advantages:
  - Simpler design
  - Simpler coding
  - Easier testing and verification
  - Flexibility
  - Reusability
  - Scalability
- Disadvantages:
  - Communication
  - More code in total

Example: Silo vs SOA

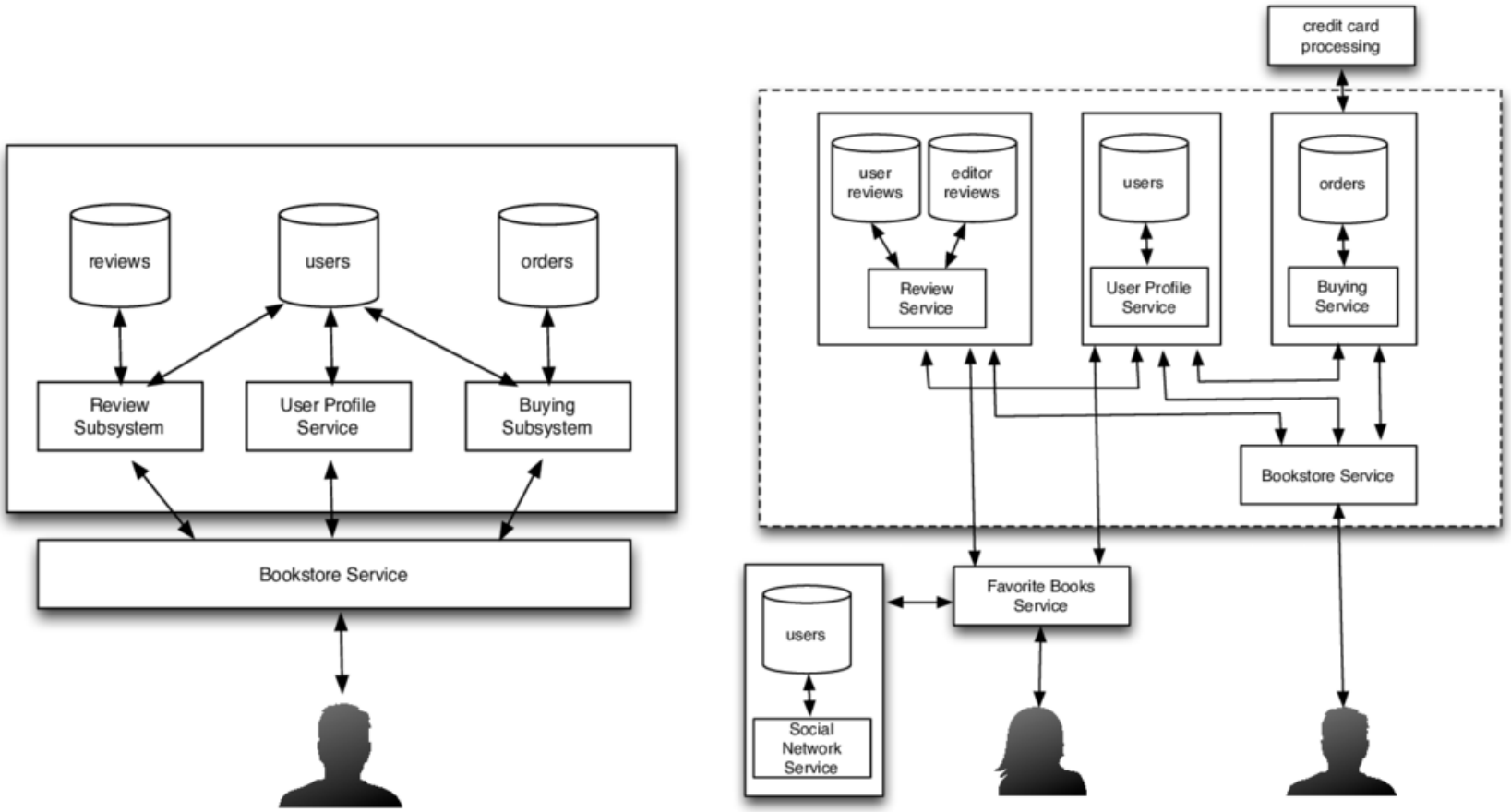

Figure 1.7 from SaaS book

SaaS Infrastructure

- Successful SaaS might serve many users and store a big amount of data
  - Amazon, Google, Microsfot, /ldots
- What infrastructure do they need? How much does it cost?
- What about smaller entrepreneurs?

SaaS Infrastructure (2)

- SaaS demands three things from its infrastructure:
  1. Communication
  2. Scalability
  3. Dependability

Clusters vs Conventional Servers

- Clusers: Simple computers connected via ethernet switches
- Much cheaper
- Scale better
- More dependable

Datacenters

- Huge clusters
- Improved commincation
- Becomes cheaper
- Infrastructure for Cloud computing
  - Examples?
- Amazon itself is a client of AWS

FarmVille - an Example of Scalability

- Hosted on AWS
- Day 4 - 1M users

FarmVille - an Example of Scalability

- Hosted on AWS
- Day 4 - 1M users
- Day 60 - 10M

FarmVille - an Example of Scalability

- Hosted on AWS
- Day 4 - 1M users
- Day 60 - 10M
- Day 270 - 75M
- What would happen if they had to depend on their own servers?

Productivity

- Most important quality of a software engineer
- Four techniques:
  1. Clarity via consiceness
  2. Code synthesis
  3. Code reuse
  4. Automation and tools

Clarity via Consiceness

- Syntax
  - `assert_greater_than_or_equal_to(a,7)`

Clarity via Consiceness

- Syntax
  - `assert_greater_than_or_equal_to(a,7)`
  - `a.should be` $\geq$ `7`

Clarity via Consiceness

- Syntax
  - `assert_greater_than_or_equal_to(a,7)`
  - `a.should be` $\geq$ `7`
- Abstraction
  - Automatic memory management
  - Higher-order functions
  - List comprehension

Code Synthesis

- Generate code from higher level specifications
  - Parser generators (i.e. GNU Bison)
  - Programming by example (i.e. Lapis)

Code Reuse

- Write code which can be reused
- Historical development
  - Procedures and functions
  - Standard libraries
  - OOP
  - Design patterns

Automation and Tools

- Automate repeating tasks
  - Save time
  - More accurate
- Concerns: Which tool to choose?
- A good software engineer must learn new technologies all the time

Exercise 2

- Clone `https://github.com/AUP-SE/ex2`