Outline

- Service Oriented Architecture (§1.7)
- Cloud Computing, Fallacies and Pitfalls (§1.8-§1.9)
- Pair Programming (§9.3)
- Ruby 101 (§3.1)
- Everything in Ruby is an Object (§3.2-§3.3)

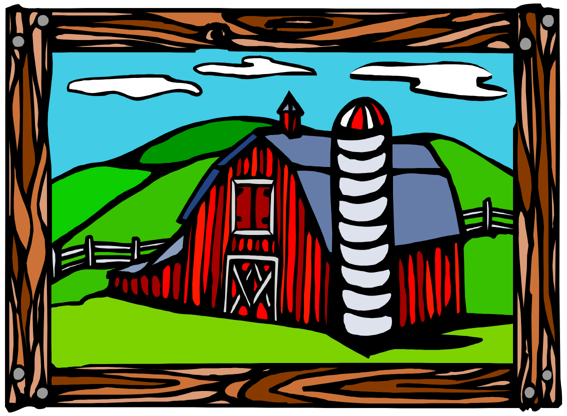
Service Oriented Architecture(SOA)

(*Engineering Long Lasting Software* §1.7)

David Patterson

Service Oriented Architecture

- SOA: SW architecture where all components are designed to be services
- Apps composed of interoperable services
  - Easy to tailor new version for subset of users
  - Also easier to recover from mistake in design
- Contrast to "SW silo" without internal APIs



CEO: Amazon shall use SOA!

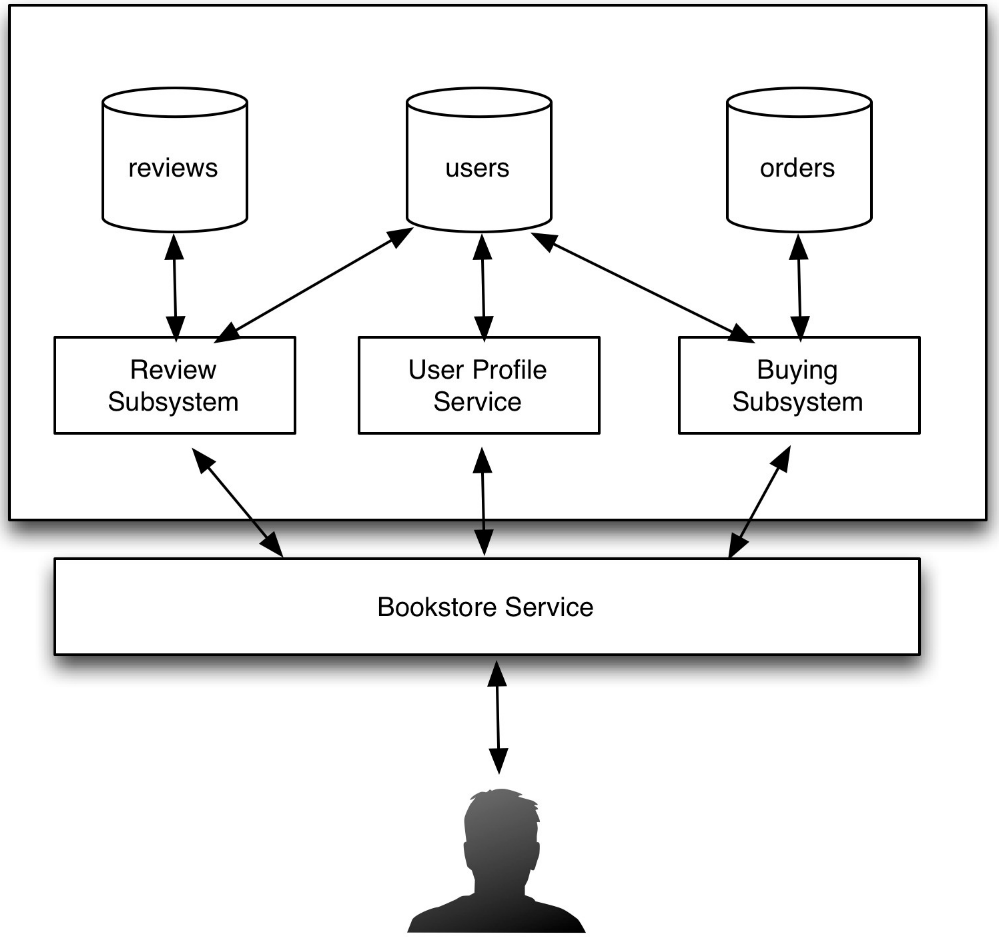
1. “All teams will henceforth expose their data and functionality through service interfaces.
2. “Teams must communicate with each other through these interfaces.
3. “There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no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

CEO: Amazon shall use SOA!

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

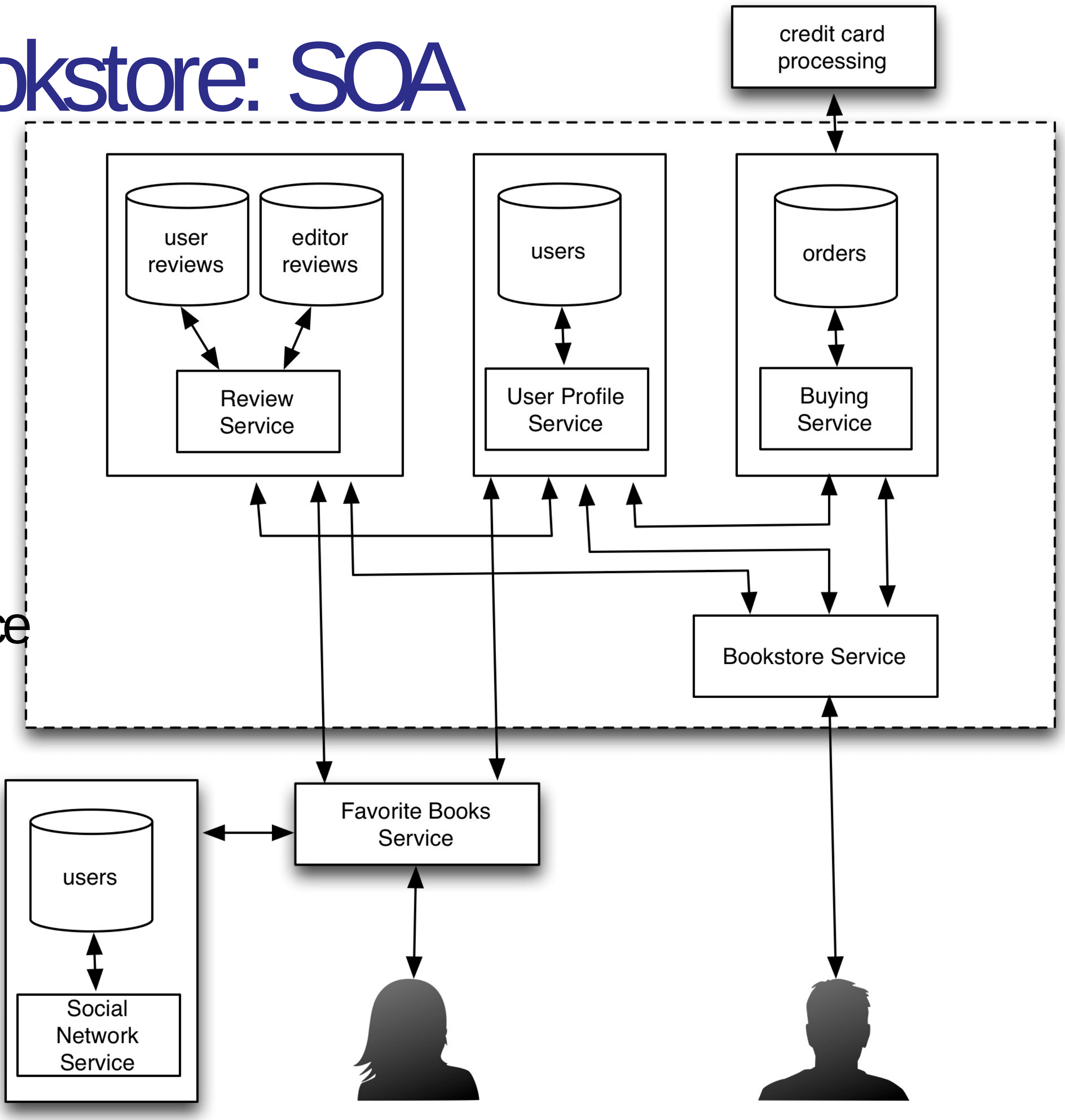
- Internal subsystems can share data directly
  - Review access user profile
- All subsystems inside single API (“Bookstore”)



(Figure 1.3, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Beta edition, 2012.)

Bookstore: SOA

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



(Figure 1.4, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Beta edition, 2012.)

Which statements NOT true about SOA?

- ☐ SOA does not affect performance
- ☐ Silo’d systems likely completely down on a failure, SOA must deal with partial failures
- ☐ SOA improves developer productivity primarily through reuse
- ☐ No service can name or access another service's data; it can only make requests for data thru an external API

Cloud Computing, Fallacies and Pitfalls, and End of Chapter 1

(*Engineering Long Lasting Software* §§1.8, 1.9, 1.12)

David Patterson

SaaS Infrastructure?

- SaaS's 3 demands on infrastructure

1. Communication: allow customers to interact with service
2. Scalability: fluctuations in demand during + new services to add users rapidly
3. Dependability: service and communication continuously available 24x7

Clusters

- Clusters: Commodity computers connected by commodity Ethernet switches

1. More scalable than conventional servers
2. Much cheaper than conventional servers
   - 20X for equivalent vs. largest servers
1. Few operators for 1000s servers
   - Careful selection of identical HW/SW
   - Virtual Machine Monitors simplify operation
1. Dependability via extensive redundancy

Warehouse Scale Computers

- Clusters grew from 1000 servers to 100,000 based on customer demand for SaaS apps
- Economies of scale pushed down cost of largest datacenter by factors 3X to 8X
  - Purchase, house, operate 100K v. 1K computers
- Traditional datacenters utilized 10% - 20%
- Make profit offering pay-as-you-go use at less than your costs for as many computers as you need

Utility Computing / Public Cloud Computing

- Offers computing, storage, communication at pennies per hour +
- No premium to scale:

```
      1000 computers @      1 hour
  =      1 computer   @ 1000 hours
```

- Illusion of infinite scalability to cloud user
  - As many computers as you can afford
- Leading examples: Amazon Web Services, Google App Engine, Microsoft Azure

2012 AWS Instances & Prices

Instance	Per Hour	Ratio to Small	Compute Units	Virtual Cores	Compute Unit/ Core	Memory (GB)	Disk (GB)	Address
Standard Small	$0.08	1.0	1.0	1	1.00	1.7	160	32 bit
Standard Large	$0.32	4.0	4.0	2	2.00	7.5	850	64 bit
Standard Extra Large	$0.64	8.0	8.0	4	2.00	15.0	1690	64 bit
High-Memory Extra Large	$0.45	5.9	6.5	2	3.25	17.1	420	64 bit
High-Memory Double Extra Large	$0.90	14.1	13.0	4	3.25	34.2	850	64 bit
High-Memory Quadruple Extra Large	$1.80	28.2	26.0	8	3.25	68.4	1690	64 bit
High-CPU Medium	$0.16	2.0	5.0	2	2.50	1.7	350	32 bit
High-CPU Extra Large	$0.66	8.0	20.0	8	2.50	7.0	1690	64 bit
Cluster Quadruple Extra Large	$1.30	15.3	33.5	16	2.09	23.0	1690	64 bit
Eight Extra Large	$2.40	28.2	88.0	32	2.75	60.5	1690	64 bit

Supercomputer for hire

- Top 500 supercomputer competition
- 532 Eight Extra Large (@ $2.40/hour), 17000 cores = 240 TeraFLOPS
- 72nd/500 supercomputer @ ~$1300 per hour
- Credit card => can use 1000s computers
- FarmVille on AWS
  - Prior biggest online game 5M users
  - What if startup had to build datacenter?
  - 4 days =1M; 2 months = 10M; 9 months = 75M

IBM Watson for Hire?

- Jeopardy Champion IBM Watson
- Hardware: 90 IBM Power 750 servers
  - 3.5 GHz 8 cores/server
- 90 @~$2.40/hour = ~$200/hour
- Cost of human lawyer or account
- For what tasks could AI be as good as highly trained person @ $200/hour?
- What would this mean for society?

Which statements NOT true about SaaS, SOA, and Cloud Computing?

- ☐ Clusters are collections of commodity servers connected by LAN switches
- ☐ The Internet supplies the communication for SaaS
- ☐ Cloud computing uses HW clusters + SW layer using redundancy for dependability
- ☐ Private datacenters could match cost of Warehouse Scale Computers if they just purchased the same type of hardware

Fallacies and Pitfalls

- Fallacy: If a software project is falling behind schedule, catch up by adding people
  - Adding people actual makes it worse!
  1. Time for new people to learn about project
  2. Communication increases as project grows, which reduces time available get work done

"Adding manpower to a late software project makes it later."

Fred Brooks, Jr. *The Mythical Man Month*

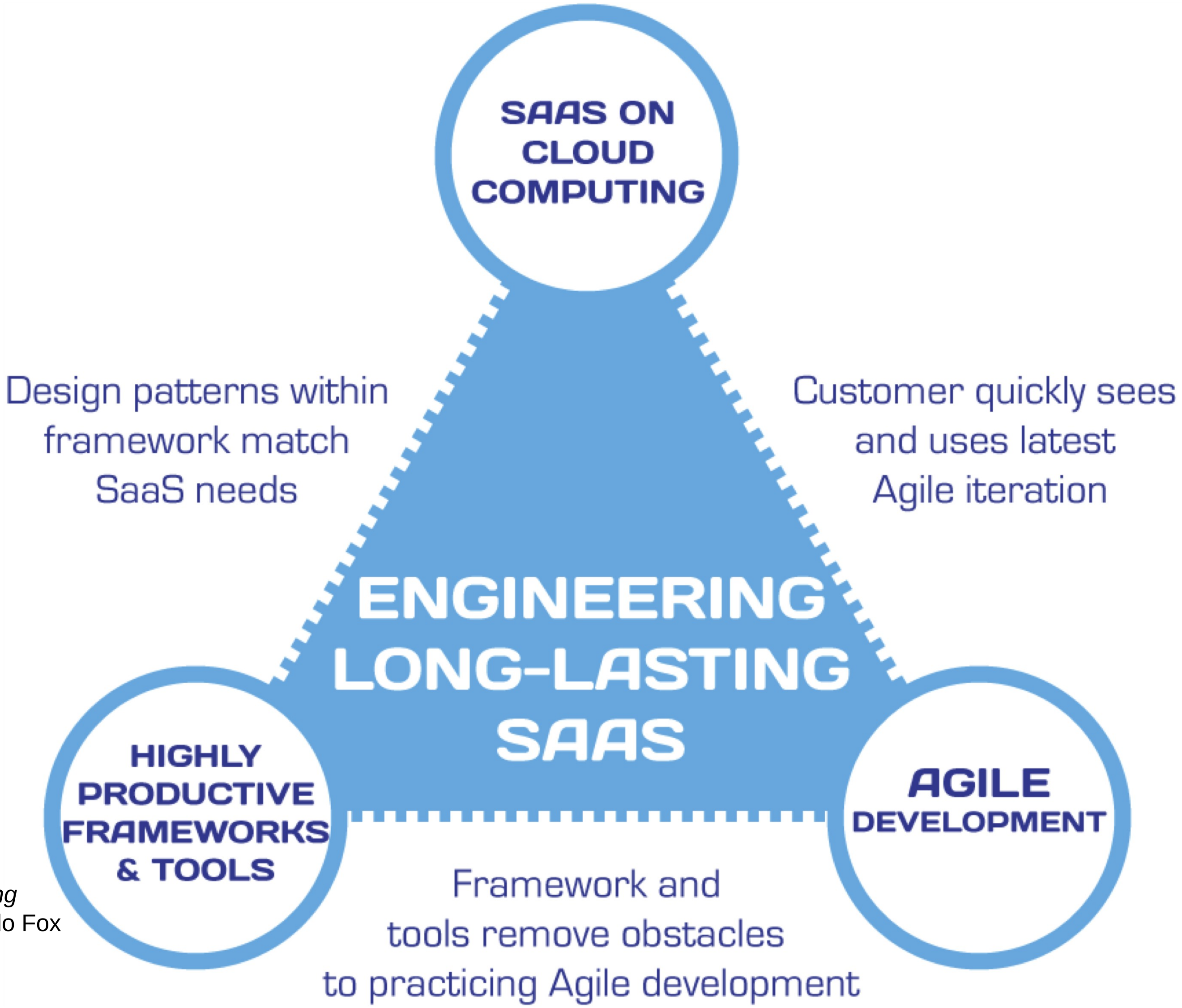
Fallacies and Pitfalls

- Fallacy: The Agile lifecycle is best for software development
  - Good match for some SW, especially SaaS
  - But not for NASA spaceshots
- Can learn SW Engineering principles in many ways, and can apply them in many ways, so pick one that matches classroom (Agile/SaaS)
- Note: you will see new lifecycles in response to new opportunities in your career, so expect to learn new ones

Fallacies and Pitfalls

- Pitfall: Ignoring the cost of software design
  - Since ≈0 cost to manufacture software, might believe ≈0 cost to remanufacture the way the customer wants
  - Ignores the cost of design and test
- (Is cost ~no cost of manufacturing software/data same rationale to pirate data? No one should pay for development, just for manufacturing?)

Summary: Engineering SW is More Than Programming


SAAS ON CLOUD COMPUTING
Design patterns within framework match SaaS needs
Customer quickly sees and uses latest Agile iteration
ENGINEERING LONG-LASTING SAAS
HIGHLY PRODUCTIVE FRAMEWORKS & TOOLS
AGILE DEVELOPMENT
Framework and tools remove obstacles to practicing Agile development

(Figure 1.7, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Beta edition, 2012.)