CS 312 Software Development

Introductions

Technology	Tools	Process
HTML JavaScript CSS Frameworks Dat:	VCS Testing frameworks Linter Continuous Integration (CI) PaaS & IaaS	Agile (Scrum) Test-driven development (TDD) DevOps Design patterns

Context: Evolving ecosystem

Shrink wrapped ⇒ Software-as-a-Service

Monolithic ⇒ Services

On-premise ⇒ Cloud

Shrink wrapped (SWS) ⇒ Software-as-a-Service (SaaS)

SWS

Client-specific binaries that must work in many HW/SW environments

+ Rich user experience

- Hard to maintain, with extensive compatibility testing required

SaaS

Online client-server model

+ One copy of SW, one HW environment (controlled by developers)

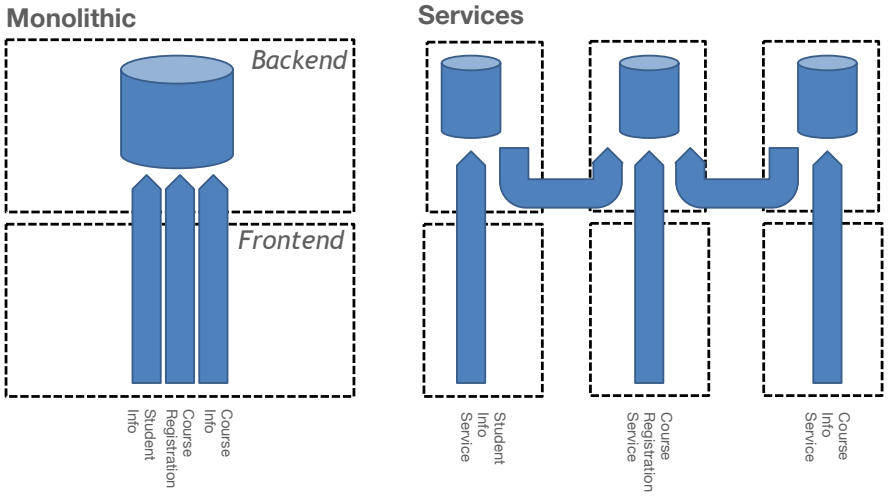
+ Easy to release updates

+ Easier to enable user collaboration

- Limited by online latency, capabilities of browser

What about mobile native applications?

Monolithic ⇒ Multiple services



Bezos' 2002 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 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.
4. It doesn't matter what technology they use. HTTP, Corba, Pubsub, custom protocols -- doesn't matter. Bezos doesn't care.
5. All 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.

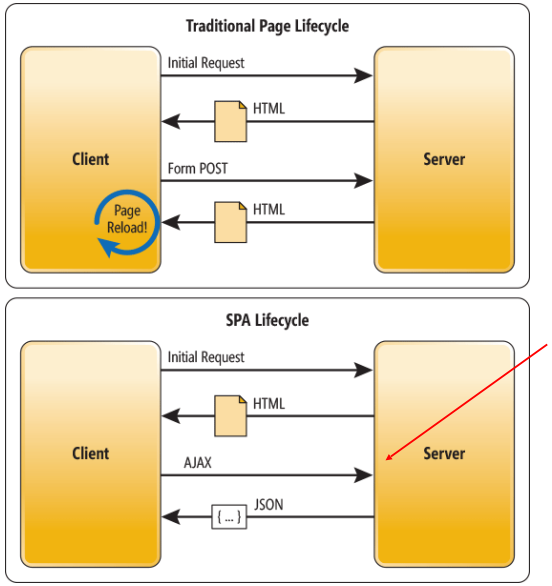
Steve Yegge blog post (2011)

SaaS 3 demands on infrastructure

1. **Communication**: Customers must be able to interact with service
2. **Scalability**: Respond to fluctuations in demand or new services adding users rapidly
3. **Dependability**: Service & communication available 24x7

Cloud providers can offer all three on a pay-as-you-go basis (utility) at hard to match prices

Single Page Applications (SPA)



Server now mostly provides data, making SPAs natural consumers of (micro)services

Wasson, Microsoft

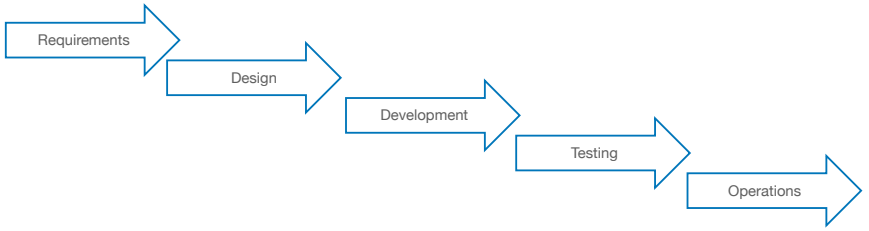
Plan & Document ⇒ Agile

"**Plan-and-Document**":

1. Before coding, the project manager makes plan
2. Write detailed documentation for all phases of the plan
3. Progress measured against the plan
4. Changes to project must be reflected in changes to documentation and the plan

Implementations: Waterfall, Spiral, …

Waterfall



Errors are caught early (and more cheaply) before manifesting in next phase
Extensive documentation is deliverable (facilitates maintenance)

Agile Manifesto (2001)

We are uncovering better ways of developing software by doing it and helping others do it.

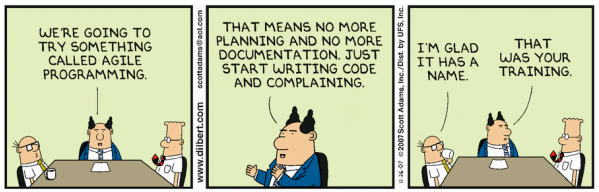
Through this work we have come to value:

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

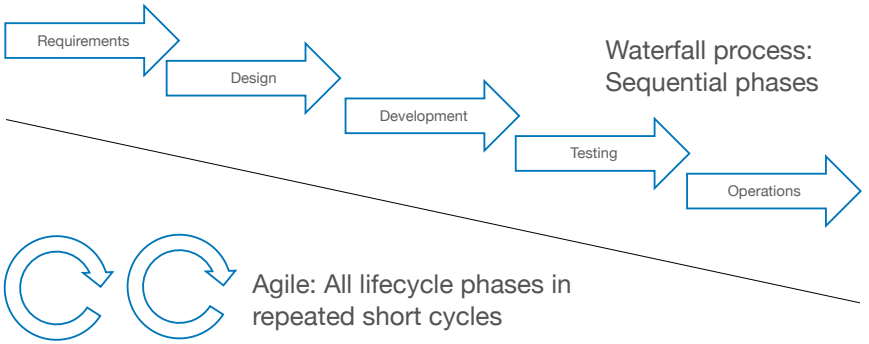
That is, while there is value in the items on the right, we value the items on the left more.

http://agilemanifesto.org

Plan & Document ⇒ Agile



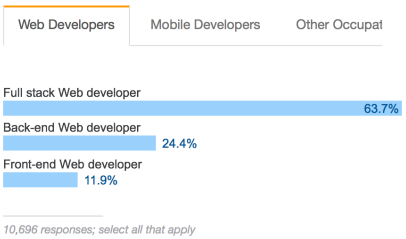
Dilbert 11/26/17



Waterfall process: Sequential phases

Agile: All lifecycle phases in repeated short cycles

"Full-Stack", "DevOps" and other buzzwords…



StackOverflow 2017 developer survey

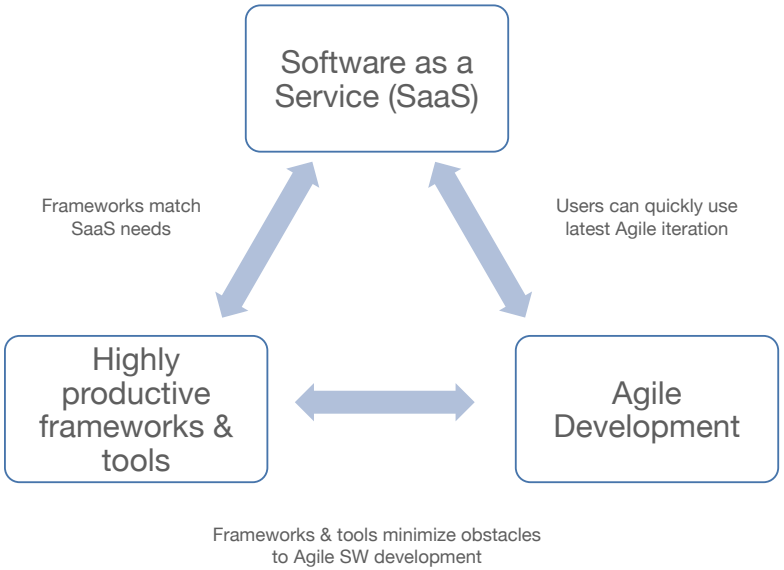
"A Full-Stack Web Developer is someone who is able to work on both the front-end and back-end portions of an application."[1]

•DevOps? Cross-functional (no more silos) teams that:

- Apply "development" practices to operations, e.g. infrastructure as code
- Automate everything
- Integrate operations into developer role

Summarizing our (the) landscape

- SW (can) evolve quickly to match user needs
- But doing so requires a development process that *embraces change*
- *Agile* is a process that embraces change (as opposed to plan & document, etc.)
- SaaS is an ideal domain for Agile processes
- Cloud gives everyone access to scalable HW and services for implementing SaaS
- SPAs are natural consumers of these (micro)services



Frameworks & tools minimize obstacles to Agile SW development

Beautiful code

Who Framed Roger Rabbit

Beautiful code:

- Meets customer needs
- Easy to evolve

The "cruft" that makes enhancements expensive is the *technical debt* created by doing the easy thing, not the "Right Thing"

"Bump the Lamp"

What I ask of you

"Do the class"

- Commit to the CS312 tools and processes
  - *Perfect practice makes perfect*
- Be a good teammate
  - *Be responsible for your learning, don't get left behind*
  - *Use your knowledge to make your team better*

Being a great teammate

TABLE 4. GREAT SOFTWARE ENGINEERS' ENGAGEMENT WITH TEAMMATES. ATTRIBUTES DISCUSSED IN DETAIL ARE IN BOLD.

Attribute and description	*Excerpt that capture interviewees' sentiment*
Creates shared context—molding another person's understanding of the situation while tailoring the message to be relevant and comprehensible to the other person.	*"Most compellingly relate the value of that abstraction as it goes to non-abstract to very abstract to each person… empathize with your audience… get them to get it."* -SDE2, Windows
Creates shared success—enabling success for everyone involved, possibly involving personal compromises.	*"Find the common good in a solution… express here's the value for you… It's a win-win situation."* -Senior Dev Lead, Windows
Creates a safe haven—creating a safe setting where engineers can learn and improve from mistakes and situations without negative consequences.	*"If you learn something from a failure, that's a wonderful sort of thing… [but not] If you're afraid of getting smacked upside the head… encourage the people to experiment, possibly succeed, possibly fail".* -Senior SDE, Office
Honest—truthful (i.e. no sugar coating or spinning the situation for their own benefit).	*"When you do make mistakes, you've got admit you made a mistake. If you try to cover up or kind of downplayed mistake, everybody will see it, it's super obvious. It affects your effectiveness."* - Partner Dev Manager, Corp Dev

Li et al. study "What Makes a Great Software Engineer"

- **Creates shared context**: molding another person's understanding of the situation while tailoring the message to be relevant and comprehensible to the other person.
- **Creates shared success**: enabling success for everyone involved, possibly involving personal compromises.
- **Creates a safe haven**: creating a safe setting where engineers can learn and improve from mistakes and situations without negative consequences.
- **Honest**: truthful (i.e. no sugar coating or spinning the situation for their own benefit).

Which of the following is a *disadvantage* of services-oriented-architecture (SOA) compared to a monolithic design? SOA:

A. May be harder to debug & tune
B. Results in lower developer productivity
C. Complexity is a poor match for small teams

Which aspect of the software lifecycle consumes the most resources?

A. Design
B. Development
C. Testing/debugging
D. Maintenance