CS5030

Agile Software Development Methods

Learning objectives

- On completing this lecture and associated reading, you should be able to
  - Understand the rationale for agile development methods
  - Differentiate between agile and plan-driven development
  - Describe agile development practices, their benefits and challenges
  - Understand the motivation for DevOps
  - Be aware of the key principles of DevOps and the support mechanisms it requires

Agile

- Oxford English Dictionary

  …

  "Business. Of a company, business activity, product, etc.:

  able to change or be changed rapidly in response to customer needs and market forces; adaptable, flexible, responsive."

Agile methods

- Based on incremental and iterative development
- Intend to deliver working software quickly and evolve it quickly to adapt to changing requirements
- Aim to reduce software overheads
  - Heavyweight plan-driven methods are not necessarily suitable for small and medium-sized projects

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."

[https://agilemanifesto.org]

Agile principles

- Customer involvement
  - Customer closely involved throughout development
- Incremental delivery
  - Software developed in increments, as defined by customer
- People, not process
  - Skills of team members should be recognised and used without overly restrictive processes
- Embracing change
  - Expect requirements to change and design for change
- Maintaining simplicity
  - In both software being developed and the development process

Plan-driven vs agile

- Plan-driven
  - Activities are planned in advance and progress is measured against this plan
- Agile
  - Incremental planning and process can be changed
- In practice
  - Processes include elements of both
  - Different process models emphasise different concerns and will be suitable for different types of systems

Agile development methods - examples

- *eXtreme Programming (XP)*
- *Scrum*
- Crystal
- Kanban
- Lean
- Feature-driven development (FDD)
- . . .

eXtreme Programming (XP)

- Developed in the late 1990s
- Extreme approach to iterative development
  - New versions may be built several times a day
  - Increments are delivered to customers every 2 weeks
  - All tests must be run for each build; a build is only accepted if all tests run successfully

XP – core practices (1)

- Planning game
  - Meeting at the start of each iteration to agree features
- On-site customer
  - Customer should be fully involved in development
- Small releases
  - First version released quickly, followed by small increments
- Simple design
  - Remove complexity

XP – core practices (2)

- Pair programming
  - 2 programmers work together on code
- Test-driven development
  - Write automated unit tests before code
- Design-improvement (refactoring)
  - Continuously improve code
- Continuous integration
  - Code changes are committed multiple times a day

XP – core practices (3)

- Collective code ownership
  - The whole team is responsible for system; anyone can change any part of code
- Coding standards
  - Team has common sets of coding practices
- Metaphor
  - Words, labels, tags or stories are applied to code elements to help communication
- Sustainable pace
  - Maintaining work-life balance with limits on overtime

XP - suitability

- Adaptive development for fast-changing requirements
- Risky projects
- Small teams
- Automated testing
- Customer participation

Scrum

- Focus on managing iterative development
  - Not on specific agile practices

- "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value."

[https://www.scrumguides.org/scrum-guide.html]

Scrum phases

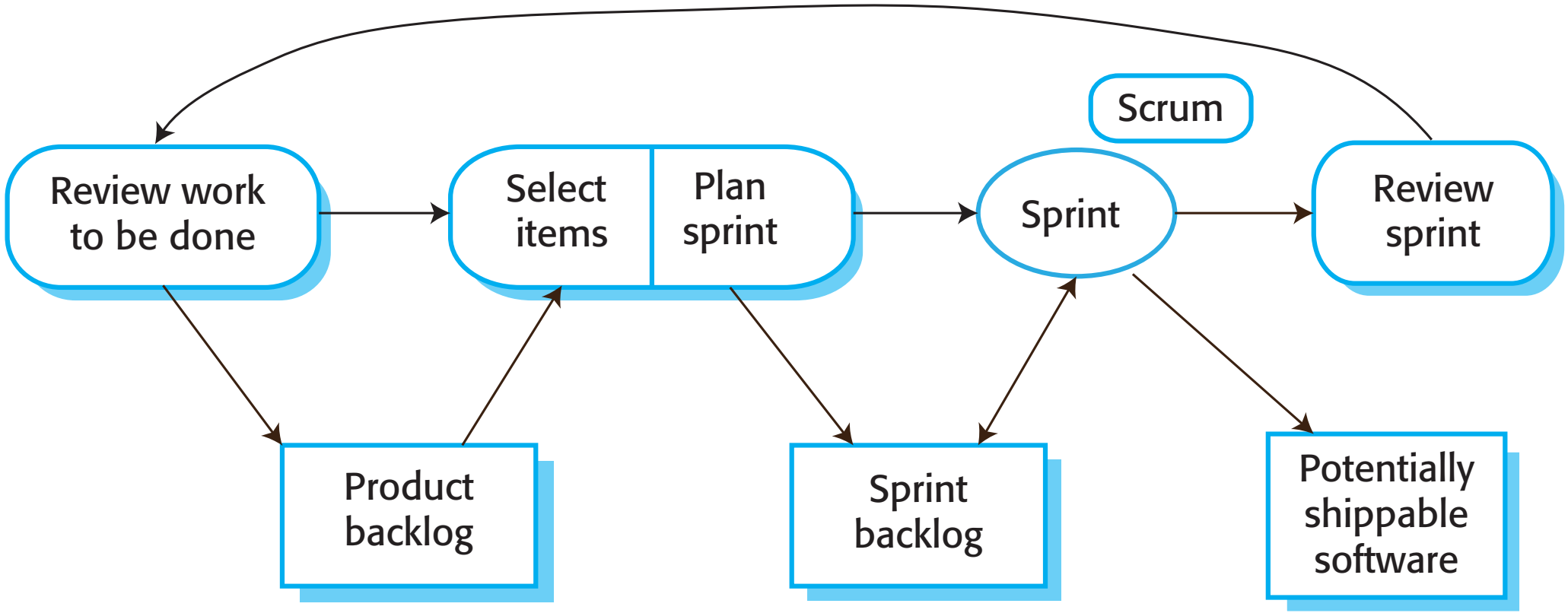
- 3 phases
  - Outline planning
    - backlog and architecture
  - Sprint cycles
    - each developing an increment of the system
  - Closure
    - wrap up and complete documentation

Scrum team

- Product owner
  - Responsible for maximising value of product created
- Development team
  - Professionals who do the work of delivering an increment
- Scrum master
  - Facilitates Scrum and liaises with external stakeholders

- Self-organising and cross-functional

Scrum sprint cycle



[Sommerville, 2016]

Scrum sprint cycle

- During development, team is isolated
  - All communications are channelled through the scrum master
- Scrum master protects team from distractions
- At the end of the sprint, completed work is reviewed and presented to stakeholders
- Next sprint cycle then begins

Scaling agile methods - challenges

- Large systems of systems, developed by different teams
- Large systems are often brownfield systems
- Configuration vs development
- Constraints from external rules and regulations
- Systems with long procurement and development time
- Diverse set of stakeholders with different perspectives and priorities

Scaling agile methods - solutions

- More up-front design and documentation
- Cross-team communication mechanisms
- Frequent system builds and regular releases
- New configuration management tools that support multi-team software development

Scaling agile methods – some solutions

- Large Scale Scrum (LeSS)
  - Team of teams
- Scaled Agile Framework (SAFe)
  - A set of organisation and workflow patterns to promote collaboration across a large number of agile teams
- Disciplined Agile Delivery (DAD)
  - Hybrid toolkit to simplify process decisions

Agility and planning

“Big design up front is dumb. Doing no design up front is even dumber”

[Dave Thomas]

Traditional software teams

- Different teams responsible for development, release and support
  - Communication delays
  - Different skills and toolsets
  - Lack of understanding of other perspectives and problems
- Result: delays in new releases and fixes reaching customers

Motivation for DevOps

- Agile practices requiring faster releases to customers
  - Avoiding bottleneck between development and deployment
- Successful examples of software services being developed and supported by the same team with improved reliability
  - For eg, Amazon
- Increasing use of software as service
  - Running on cloud

DevOps

- Development + Operations
- Integration of development, deployment and support
- Cross-functional teams work on continuous delivery of operational features
  - Software engineering, user interface design, security engineering, customer interaction, etc
  - Culture of mutual respect, sharing and learning
- DevOps concept is interpreted and implemented differently by different organisations
  - Dependent on software delivery mechanism, etc

DevOps principles

- Everyone (in the team) is responsible for everything
  - Development, release & support
- Everything that can be automated should be automated
  - Minimal manual involvement in software deployment
- Measure first, change later
  - Data collected about system and its operation to help decision making

DevSecOps

- Development + Security + Operations

- Security is a shared responsibility throughout the software lifecycle
  - including management

Key points

- Agile methods are oriented to a different set of concerns compared to traditional methods
- Agile methods are enabled by recent software engineering practices and resources, including computers, networking and storage
- They reduce some risks but could potentially introduce others
  - Always a tradeoff
- Agile methods are not incompatible with planning and design
- Agile methods are not silver bullets
- DevOps is the integration of software development and management of software after deployment
- Automation, source code management and measurement are key aspects of DevOps