SOFTWARE ENGINEERING

CO3001

CHAPTER 10 – AGILE SOFTWARE DEVELOPMENT

Anh Nguyen-Duc
Tho Quan

WEEK 10

Adapted from https://iansommerville.com/software-engineering-book/slides/

TOPICS COVERED

- ✓ Agile methods
- ✓ Scrum in nutshell
- ✓ Other Agile approaches

RAPID SOFTWARE DEVELOPMENT

- ✓ Rapid development and delivery is now often the most important requirement for software systems
  - Businesses operate in a fast-changing requirement
  - => practically impossible to have stable software requirements
  - Software has to evolve quickly to reflect changing business needs.
  - Plan-driven development (waterfall, incremental dev.) is essential for some types of system but does not meet these business needs.

AGILE DEVELOPMENT

- emerged in the late 1990s
- aim to radically reduce the delivery time for working software systems

- ✓ Program specification, design and implementation are inter-leaved
  - The system = a series of versions / increments
  - Stakeholders involved in version specification and evaluation
  - Frequent delivery of new versions for evaluation

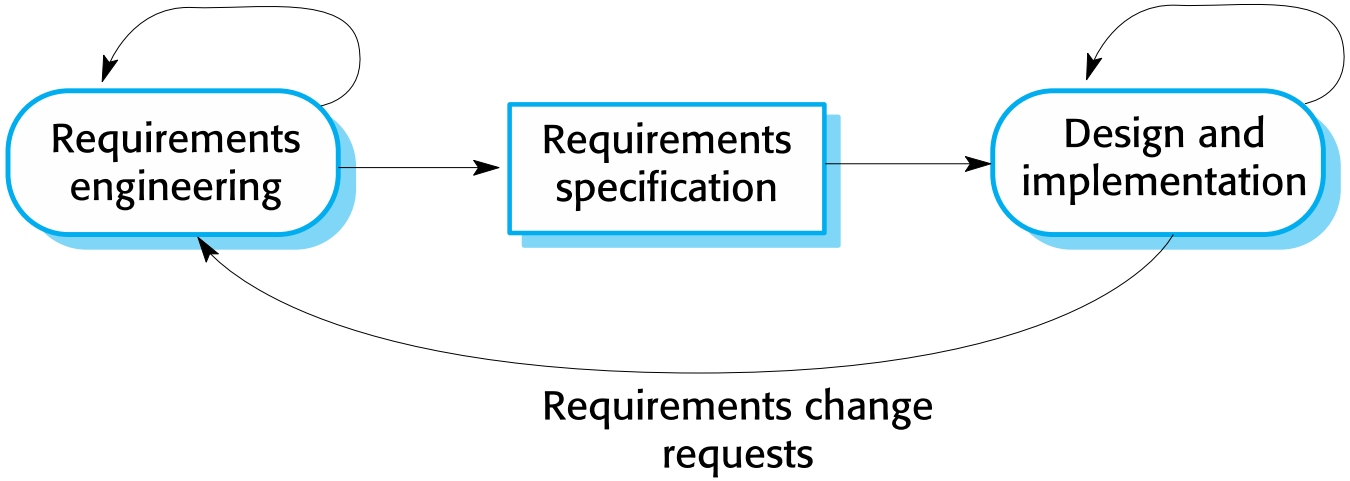
- ✓ Minimal documentation – focus on working code

  - Extensive tool support (e.g. automated testing tools) used to support development.

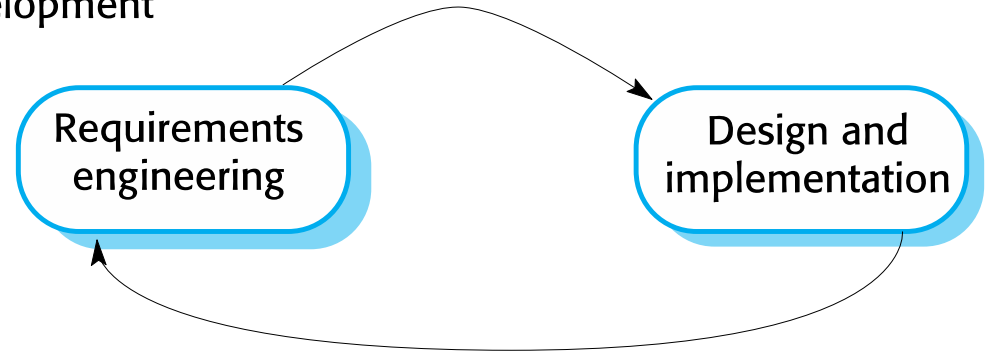
PLAN-DRIVEN AND AGILE DEVELOPMENT

Plan-based development

i.e.: waterfall model, incremental development



Agile development



AGILE METHODS

THE AGILE MANIFESTO

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

Source: www.agilemanifesto.org

VALUE 1: INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS

- **Strong players**: a must, but can fail if don't work together.
- **Strong player:** not necessarily an 'ace;' work well with others!
  - Communication and interacting is **more important** than raw talent.
- **'Right' tools** are vital to smooth functioning of a team.
- **Start small**. Find a free tool and use until you can demo you've outgrown it. Don't assume bigger is better. Start with white board; flat files before going to a huge database.
- **Building a team** more important than **building environment.**
  - Some managers build the environment and expect the team to fall together.
  - Doesn't work.
  - Let the team build the environment on the **basis of need**.

VALUE 2: WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION

- **Code** – not ideal medium for communicating rationale and system structure.
  - Team needs to produce human readable documents describing system and design decision rationale.
- **Too much documentation is worse than too little**.
  - Take time; more to keep in sync with code; Not kept in sync? it is a lie and misleading.
- **Short rationale and structure document.**
  - Keep this in sync; Only highest level structure in system kept.
  - **Fatal flaw**: Pursue documentation instead of software:
- **Rule**: Produce no document unless need is immediate and significant.

VALUE 3: CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION (1 OF 2)

- Not possible to describe software requirements up front and leave someone else to develop it within cost and on time.
- Customers **cannot** just cite needs and go away
- Successful projects require **customer feedback on a regular and frequent basis** – and not dependent upon a contract or SOW.

VALUE 3: CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION (2 OF 2)

- **Best contracts are NOT** those specifying requirements, schedule and cost.
  - Become meaningless shortly.
- **Far** better are **contracts that govern the way the development team and customer will work together.**
- Key is intense collaboration with customer and a contract that governed collaboration rather than details of scope and schedule
  - Details ideally **not** specified in contract.
  - Rather contracts could pay when a block (deliverable) passed customer's acceptance tests.
  - With frequent deliverables and feedback, **acceptance tests** never an issue.

VALUE 4: RESPONDING TO CHANGE OVER FOLLOWING A PLAN

- Our plans and ability to respond to changes is critical!
- Course of a project cannot be predicted far into future.
  - Too many variables; not many good ways at estimating cost
  - As developers gain knowledge of the system and as customer gains knowledge about their needs, some tasks will become unnecessary.
- Better planning strategy: – make detailed plans for the next few weeks, very rough plans for the next few months, and extremely crude plans beyond that.
- Need to know what we will be working on the next few weeks; roughly for the next few months; a vague idea what system will do after a year.
- Only invest in a detailed plan for immediate tasks; once plan is made, difficult to change due to momentum and commitment.
  - The lower resolution parts of the plan can be changed with relative ease.

INTRODUCTION TO SCRUM

SCRUM HAS BEEN USED BY:

23
Shares

Now, let's demystify some of these terms...

Key elements of the Spotify model

The Spotify model is centered around simplicity. When Spotify began organizing around their work, they identified a handful of important elements on how people and teams should be structured.

Squads

Similar to a scrum team, Squads are cross-functional, autonomous teams (typically 6-12 individuals) that focus on one feature area. Each Squad has a unique mission that guides the work they do, an agile coach for support, and a

Teaching Scrum at Tesla, Talking with Silicon Valley Agile
ship Network

d | Aug 30, 2017 | Blog | 7 comments

During the week of the Tesla Model 3 Launch in Fremont, CA, Tesla invited Scrum Inc. to host a Scrum Inc Scrum Master and Scrum Inc Product Owner course at the factory. On Wednesday, we had a tour at the production site in Elon Musk's personal golf cart. An awesome time was had by all!

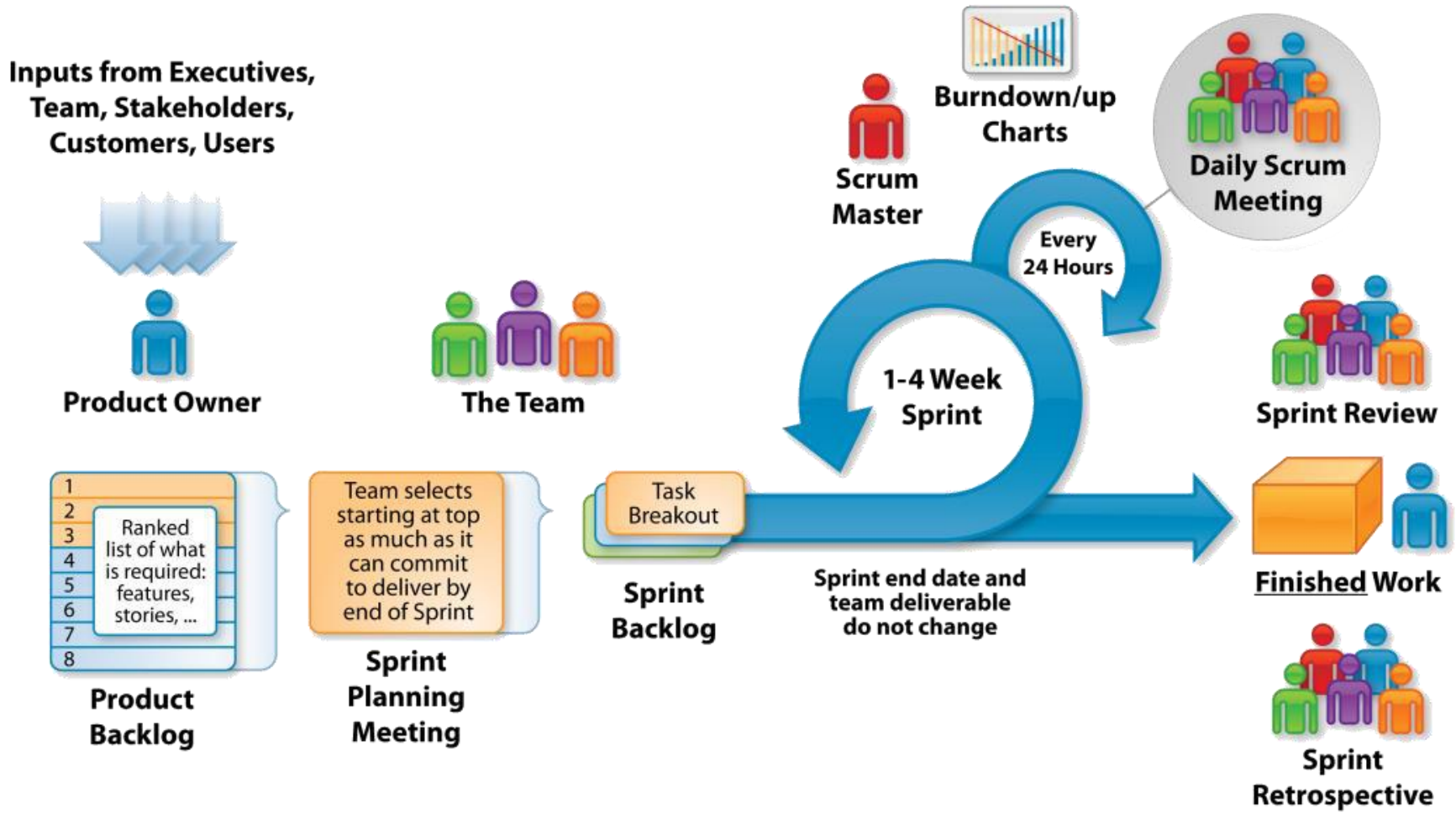
- Salesforce.com
- Time Warner
- Turner Broadcasting
- Oce
- …

SCRUM HAS BEEN USED FOR:

- Commercial software
- In-house development
- Contract development
- Fixed-price projects
- Financial applications
- ISO 9001-certified applications
- Embedded systems
- 24x7 systems with 99.999% uptime requirements
- Video game development
- FDA-approved, life-critical systems
- Satellite-control software
- Websites
- Handheld software
- Mobile phones
- Network switching applications
- ISV applications

HOW SCRUM WORKS

https://www.youtube.com/watch?v=XJeaPyWOCaw



SCRUM FRAMEWORK

Roles

- Product owner
- ScrumMaster
- Team

Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

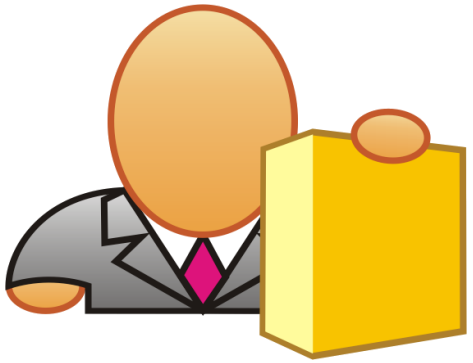
SCRUM

SCRUM ROLES

Roles

- Product owner
- ScrumMaster
- Team

PRODUCT OWNER



- ✓ Define the features of the product
- ✓ Decide on release date and content
- ✓ Be responsible for the profitability of the product (ROI)
- ✓ Prioritize features according to market value
- ✓ Adjust features and priority every iteration, as needed
- ✓ Accept or reject work results

SCRUM MASTER

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

THE TEAM

- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints

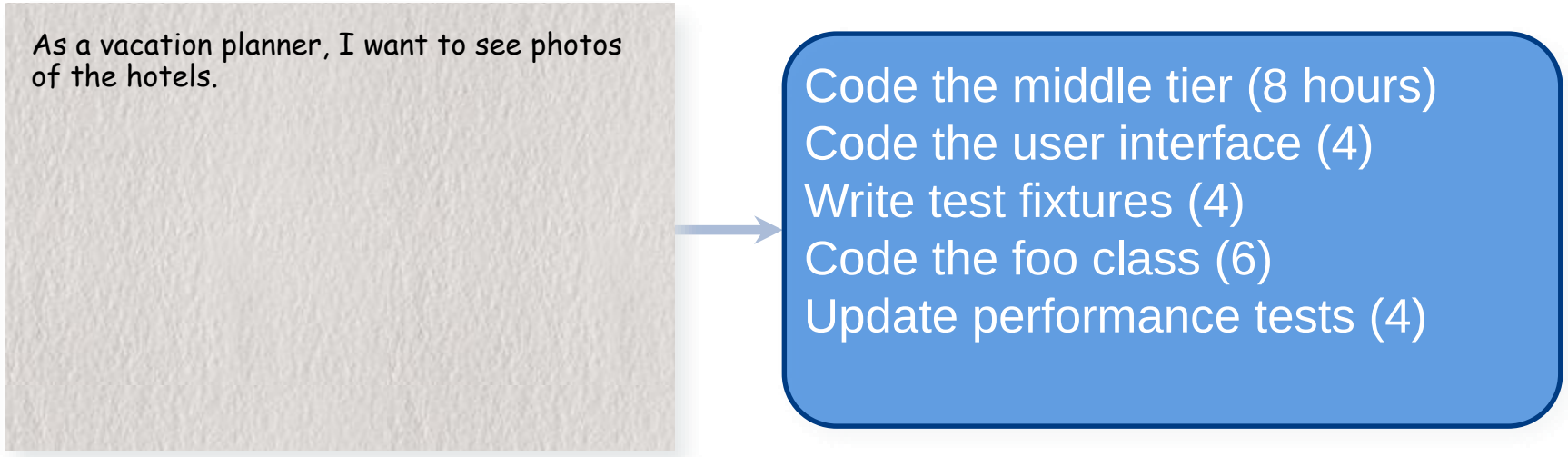
SCRUM CEREMONIES

Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

SPRINT PLANNING

- Decide how to achieve sprint goal (design)
- Team selects items from the product backlog they can commit to completing
- Create sprint backlog (tasks) from product backlog items (user stories / features)
- Estimate sprint backlog in hours. Tasks is estimated (1-16 hours)
- Collaboratively, not done alone by the ScrumMaster

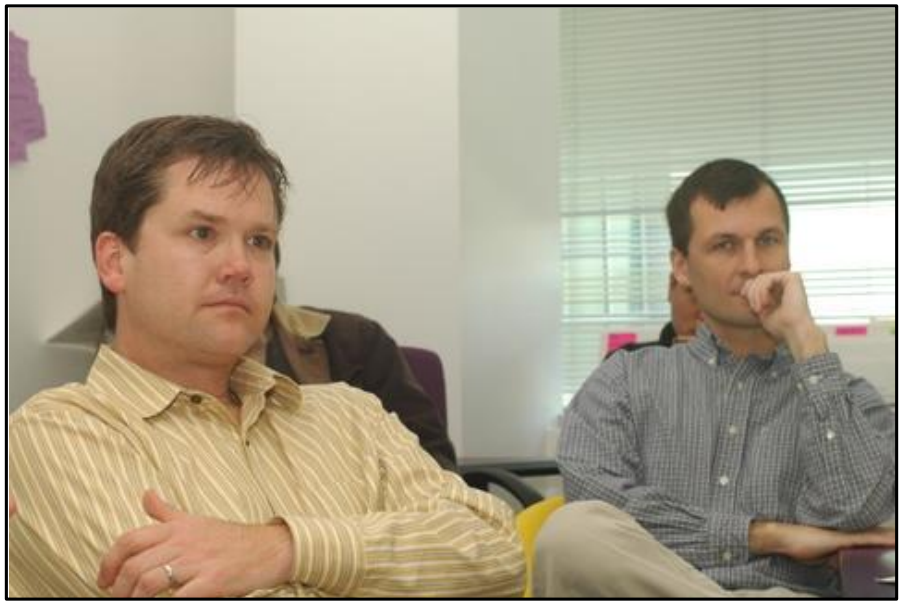


THE DAILY SCRUM (YOUR VERSION: BI/WEEKLY SCRUM)

- Daily, 15-minutes, stand-up
- Not for problem solving
- Moderated by a Scrum Master
- Three questions:
  - What did you do yesterday?
  - What will you do today?
  - Is anything in your way?

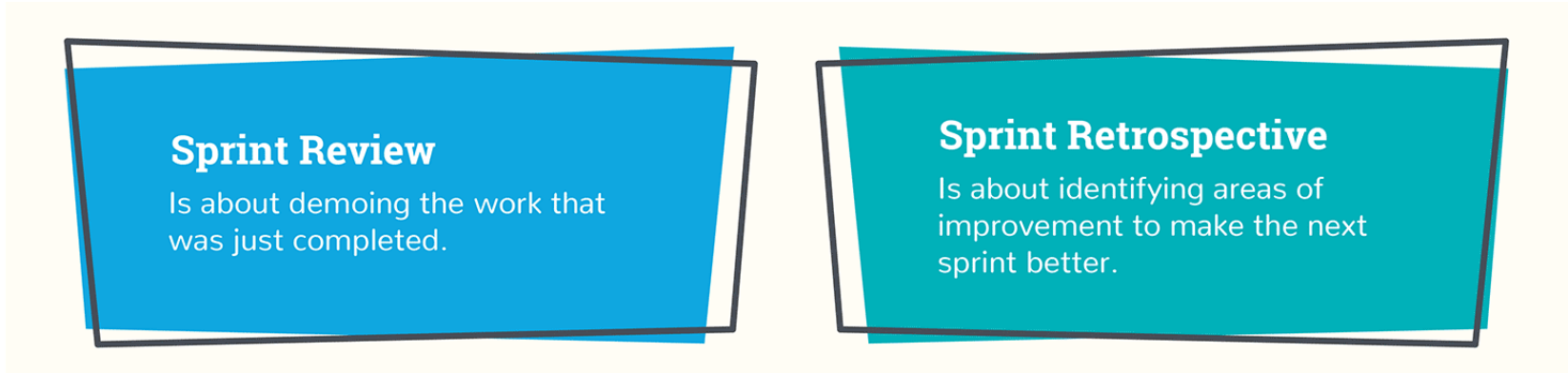
THE SPRINT REVIEW MEETING (OUR FORMAL SUPERVISION MEETING)

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
  - No slides
- Whole team participates
- Invite the world



SPRINT RETROSPECTIVE MEETING

- ✓ Periodically take a look at what is and is not working
- ✓ Typically 15–30 minutes
- ✓ Done after every sprint
- ✓ Whole team participates
- ✓ Moderated by a Scrum Master
- ✓ Each team member is asked to identify specific things that the team should:
  - Start doing
  - Stop doing
  - Continue doing



IN-CLASS EXERCISE

Reflect on tools, collaboration, teamwork, meetings, etc. you have done for Delivery 2. Perform the retrospective within your team:

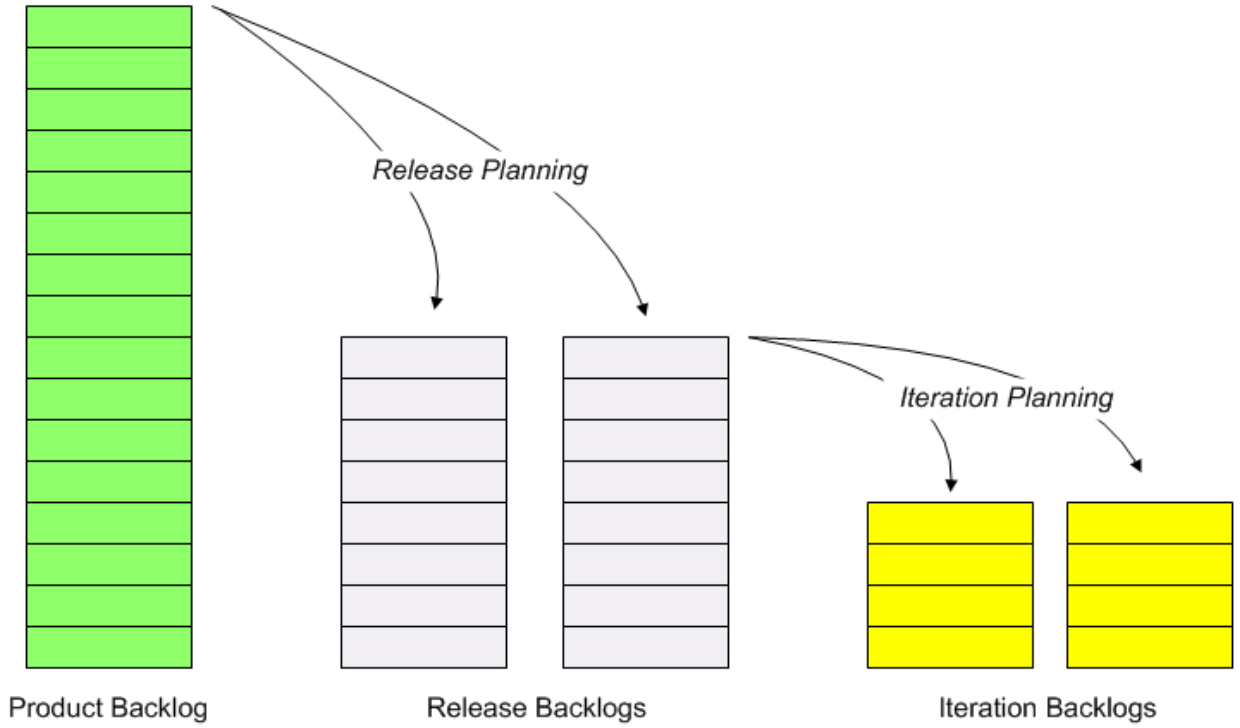
- One team member act as a Scrum Master
- The Scrum master asks each team member about what he/ she will:
  - Start doing
  - Stop doing
  - Continue doing

SCRUM ARTIFACTS

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

PRODUCT BACKLOG VS. SPRINT BACKLOG



USER STORIES

- ✓ An informal, natural language description of one or more features of a software system
- ✓ Written from the perspective of an end user
- ✓ A common pattern:
  - ***As a <role> I can/ want <capability>, so that <receive benefit>***

USER STORIES IN SPRINT BACKLOG VS PRODUCT BACKLOG

	Product BackLog	Sprint BackLog
Level of detail	Less detailed	Very detailed
Item	User Story	Task
Estimation Units	Story Points	Hours
Doc owner	Product Owner	Team
Revised	Weekly	Daily
Duration	Project	Sprint
Workbook	Product Backlog	Iteration Backlog

EFFORT ESTIMATION

- In the beginning of the Sprint, we should be able to tell how long/ how costly it will take to accomplish User Story ABC
  - “I think it will take me five days full-time to finish the User Story ABC”
  - “This week I have a midterm exam, so I take me all together seven days…”
  - “This User Story ABC might need 3000 Line of Codes…”
  - “This User Story ABC will have 3 user interfaces…”
  - “ I would give it 5 out of 10 points in term of efforts needed”
- Many different approaches for estimation
- No estimation is perfect
- What is important is to have the consistent understanding about the effort/ complexity of the task/ the user story throughout the WHOLE team

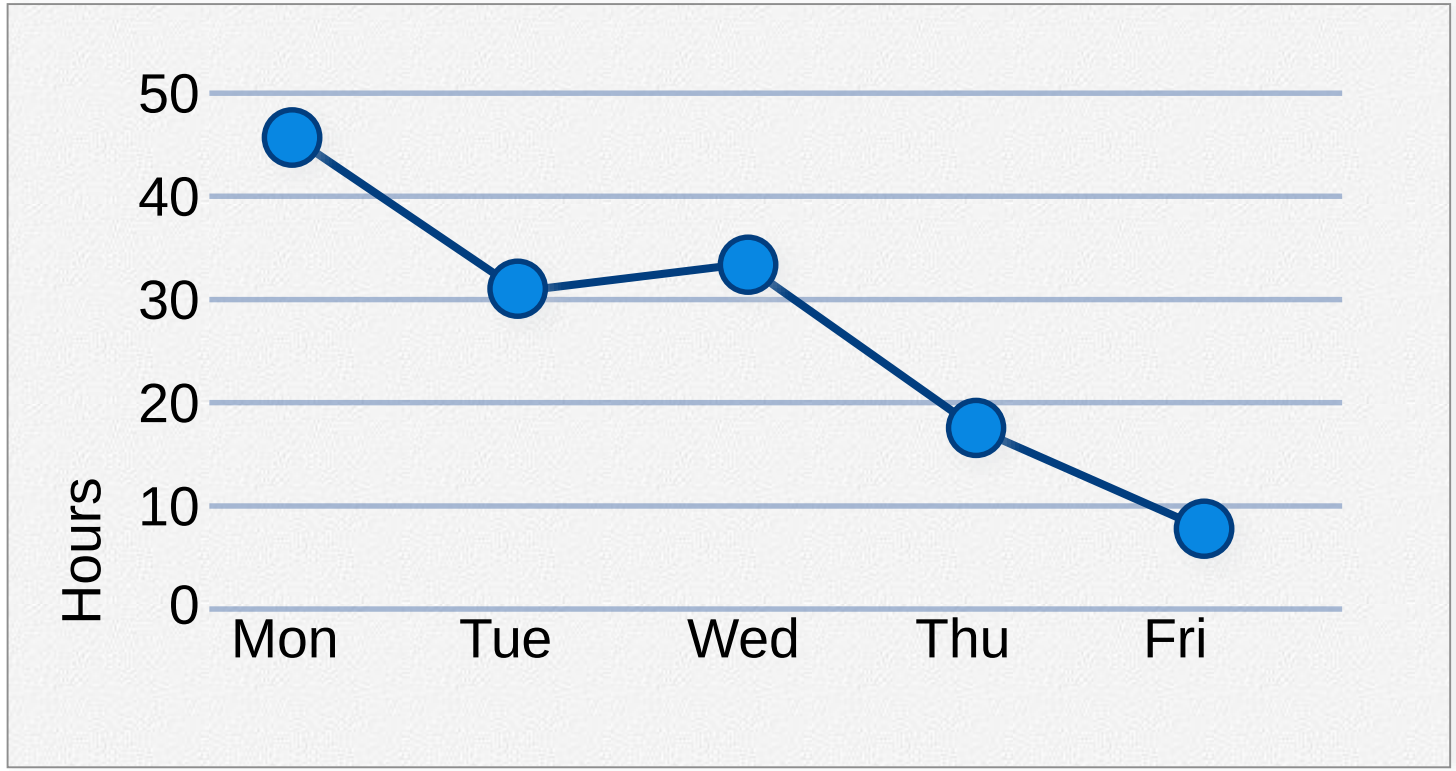
PLANNING POKER

PLANNING POKER

- ✓ For each user story, a team member can give a number 0, 1, 2, 3, 5, 8, 13, and 21 (A story estimated as a 2 should be about one fourth as difficult as a story estimated as an 8)
  - 0 is the most simple/ easiest
  - 21 is the most difficult/ most effort needed
- ✓ A moderator ask each member:
  - Read aloud the number
- ✓ The team members with the lowest and highest estimates explain why they chose their scores
- ✓ Discuss until the team reach consensus on one number

BURNDOWN CHART

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				



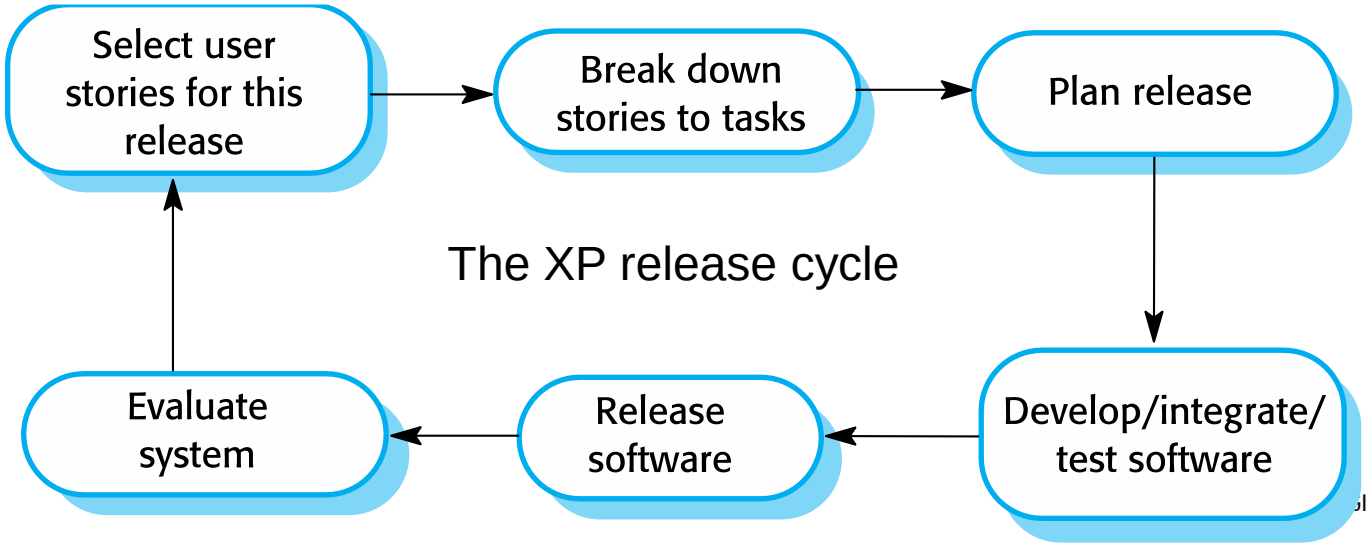
SCRUM BENEFITS

- ✓ The product is broken down into a set of manageable and understandable chunks.
- ✓ Unstable requirements do not hold up progress.
- ✓ The whole team have visibility of everything and consequently team communication is improved.
- ✓ Customers see on-time delivery of increments and gain feedback on how the product works.
- ✓ Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

EXTREME PROGRAMMING

late 1990s

- ✓ Extreme Programming (XP) takes an 'extreme' approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.



The XP release cycle