SOFTWARE ENGINEERING

CO3001

CHAPTER 3 – AGILE SOFTWARE DEVELOPMENT

WEEK 12

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

TOPICS COVERED

- ✓ Agile methods
- ✓ Agile development techniques
- ✓ Agile project management

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

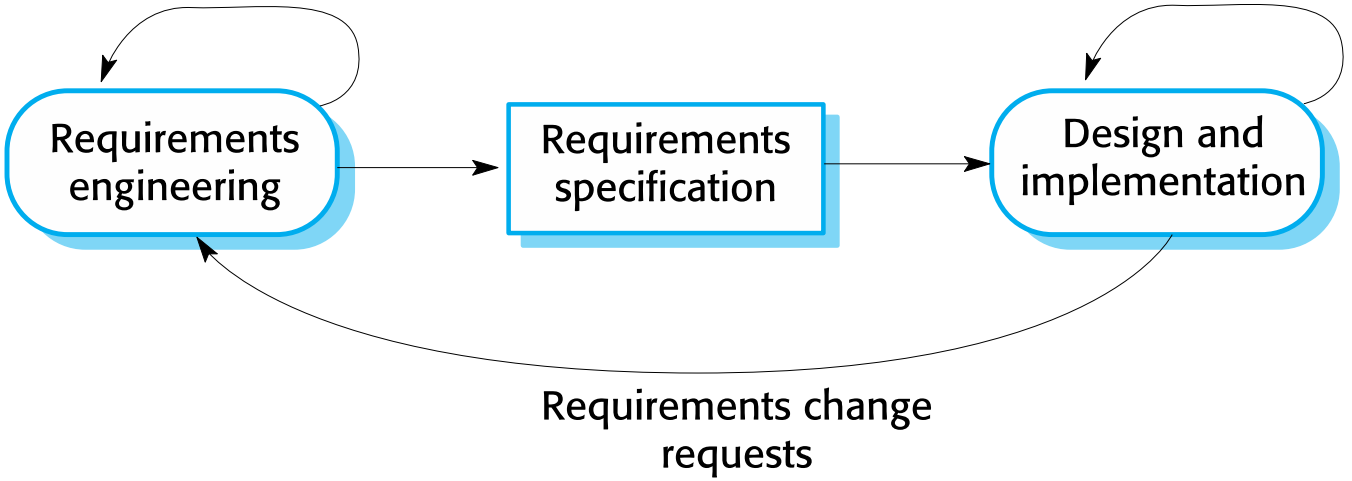
- ✓ Minimal documentation – focus on working code

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

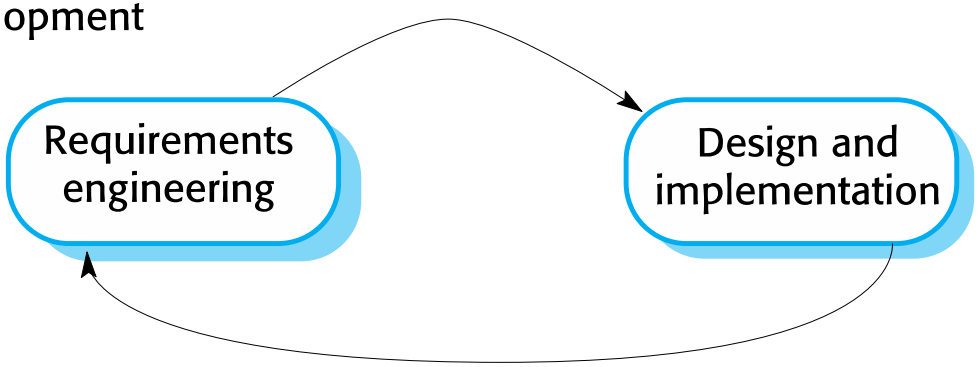
PLAN-DRIVEN AND AGILE DEVELOPMENT

Plan-based development

i.e.: waterfall model, incremental development



Agile development



AGILE METHODS

AGILE METHODS

- ✓ Agile methods:
  - Focus on the code rather than the design
  - Are based on an iterative approach to software development
  - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.

- ✓ Aims:
  - to reduce overheads in the software process (e.g. by limiting documentation)
  - to respond quickly to changing requirements without excessive rework.

THE PRINCIPLES OF AGILE METHODS

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental **delivery**	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People, not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace **change**	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain **simplicity**	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

AGILE METHOD APPLICABILITY

- ✓ For a small or medium-sized product for sale.
  - Virtually all software products and apps are now developed using an agile approach

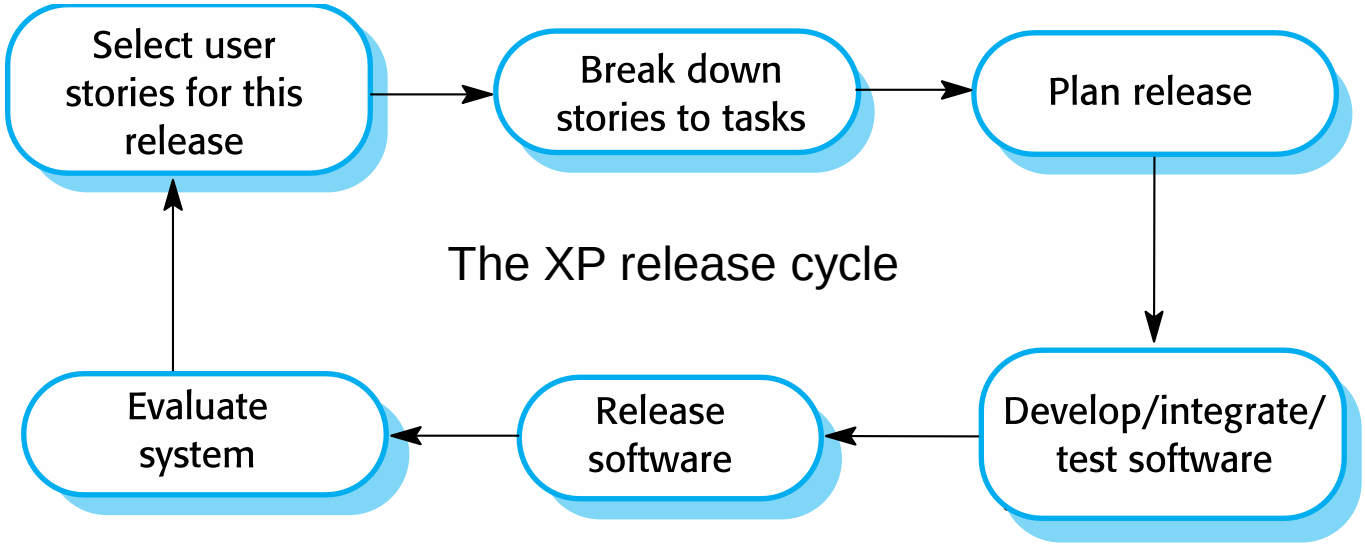
- ✓ Clear commitment from the customer to become involved
  - in the development process
  - and where there are few external rules and regulations that affect the software.

AGILE DEVELOPMENT TECHNIQUES

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

EXTREME PROGRAMMING PRACTICES

Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

XP AND AGILE PRINCIPLES

- ✓ Incremental development
  - supported through small, frequent system releases.
- ✓ Customer involvement
  - means full-time customer engagement with the team.
- ✓ People not process
  - through pair programming, collective ownership and a process that avoids long working hours.
- ✓ Change supported
  - through regular system releases.
- ✓ Maintaining simplicity
  - through constant refactoring of code.

INFLUENTIAL XP PRACTICES

- Extreme programming has a technical focus and is not easy to integrate with management practice in most organizations.
- Consequently, while agile development uses practices from XP, the method as originally defined is not widely used.

✓ Key practices

- User stories for specification
- Refactoring
- Test-first development
- Pair programming

USER STORIES FOR REQUIREMENTS

- In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.
- User requirements are expressed as user stories or scenarios.
- These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.
- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

A 'PRESCRIBING MEDICATION' STORY

Prescribing medication

Kate is a doctor who wishes to prescribe medication for a patient attending a clinic. The patient record is already displayed on her computer so she clicks on the medication field and can select 'current medication', 'new medication' or 'formulary'.

If she selects 'current medication', the system asks her to check the dose; If she wants to change the dose, she enters the new dose then confirms the prescription.

If she chooses 'new medication', the system assumes that she knows which medication to prescribe. She types the first few letters of the drug name. The system displays a list of possible drugs starting with these letters. She chooses the required medication and the system responds by asking her to check that the medication selected is correct. She enters the dose then confirms the prescription.

If she chooses 'formulary', the system displays a search box for the approved formulary. She can then search for the drug required. She selects a drug and is asked to check that the medication is correct. She enters the dose then confirms the prescription.

The system always checks that the dose is within the approved range. If it isn't, Kate is asked to change the dose.

After Kate has confirmed the prescription, it will be displayed for checking. She either clicks 'OK' or 'Change'. If she clicks 'OK', the prescription is recorded on the audit database. If she clicks on 'Change', she reenters the 'Prescribing medication' process.

EXAMPLES OF TASK CARDS FOR PRESCRIBING MEDICATION

Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

REFACTORING

- Conventional wisdom: design for change, anticipating changes => reduce cost later in life cycle
- XP: constant code improvement (refactoring) to make changes easier when they have to be implemented.

- ✓ Make possible software improvements
  - even where there is no immediate need for them.
- ✓ Refactoring advantages:
  - Improves the understandability of the software => reduces the need for documentation.
  - Code is well-structured and clear => changes are easier to make
- ✓ Disadvantages:
  - some changes requires architecture refactoring => expensive.

EXAMPLES OF REFACTORING

- ✓ Re-organization of a class hierarchy to remove duplicate code.
- ✓ Tidying up and renaming attributes and methods to make them easier to understand.
- ✓ The replacement of inline code with calls to methods that have been included in a program library.

TEST-FIRST DEVELOPMENT

- Testing is central to XP and XP has developed an approach where the program is tested after every change has been made.

✓ XP testing features:

- Test-first development.
- Incremental test development from scenarios.
- User involvement in test development and validation.
- Automated test harnesses are used to run all component tests each time that a new release is built.

TEST-DRIVEN DEVELOPMENT

- ✓ Writing tests before code clarifies the requirements to be implemented.
- ✓ Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly.
  - Usually relies on a testing framework such as JUnit.
- ✓ All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.

CUSTOMER INVOLVEMENT IN TESTING

- The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system.
- The customer who is part of the team writes tests as development proceeds. All new code is therefore validated to ensure that it is what the customer needs.
- However, people adopting the customer role have limited time available and so cannot work full-time with the development team. They may feel that providing the requirements was enough of a contribution and so may be reluctant to get involved in the testing process.

TEST CASE DESCRIPTION FOR DOSE CHECKING

Test 4: Dose checking

Input:
1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:
1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:
OK or error message indicating that the dose is outside the safe range.

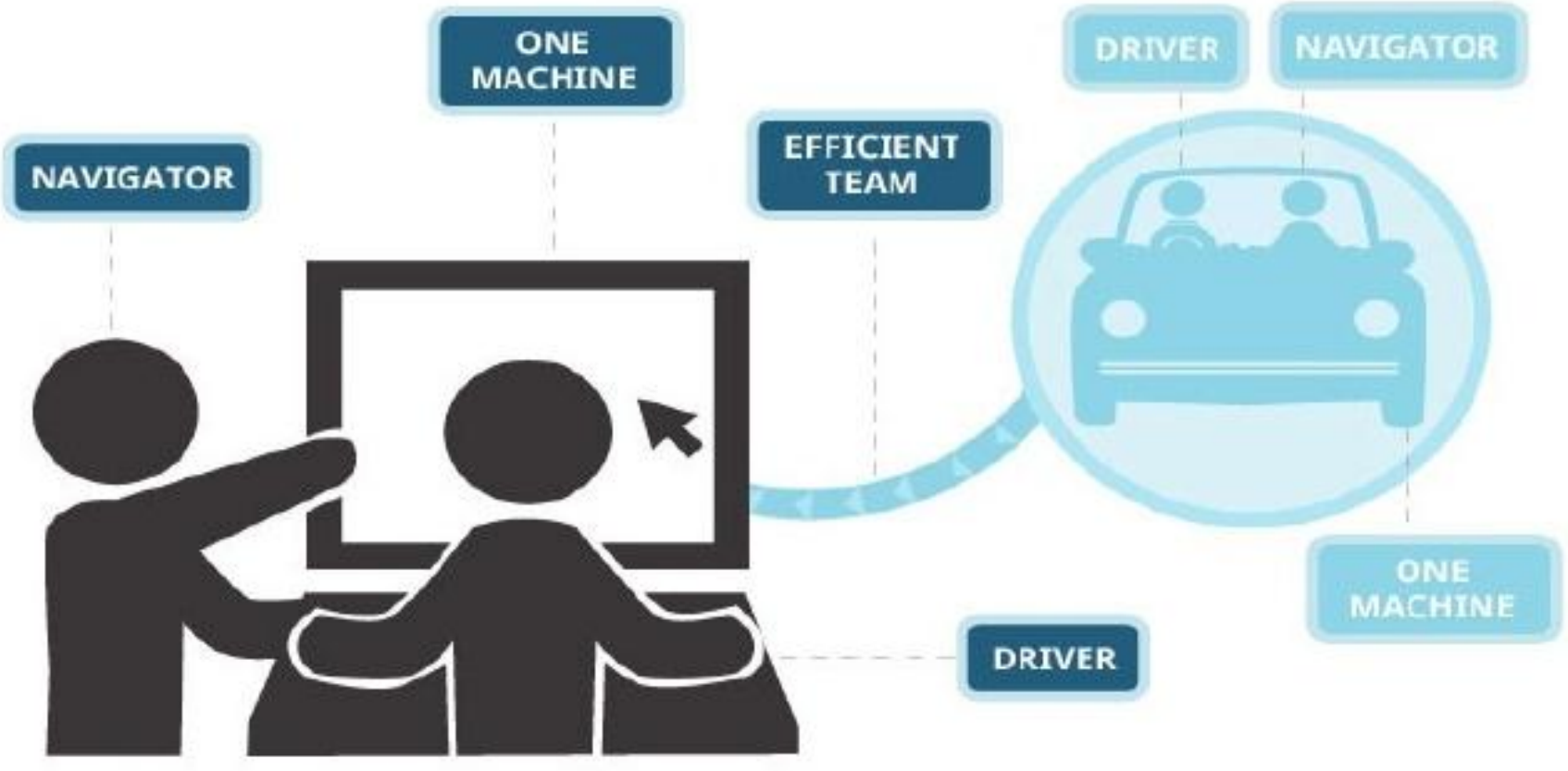
TEST AUTOMATION

- ✓ Test automation means that tests are written as executable components before the task is implemented
  - These testing components should be stand-alone, should simulate the submission of input to be tested and should check that the result meets the output specification. An automated test framework (e.g. JUnit) is a system that makes it easy to write executable tests and submit a set of tests for execution.
- ✓ As testing is automated, there is always a set of tests that can be quickly and easily executed
  - Whenever any functionality is added to the system, the tests can be run and problems that the new code has introduced can be caught immediately.

PROBLEMS WITH TEST-FIRST DEVELOPMENT

- ✓ Programmers prefer programming to testing and sometimes they take short cuts when writing tests. For example, they may write incomplete tests that do not check for all possible exceptions that may occur.
- ✓ Some tests can be very difficult to write incrementally. For example, in a complex user interface, it is often difficult to write unit tests for the code that implements the 'display logic' and workflow between screens.
- ✓ It difficult to judge the completeness of a set of tests. Although you may have a lot of system tests, your test set may not provide complete coverage.

PAIR PROGRAMMING



PAIR PROGRAMMING

- ✓ Working in pairs, developing code together.
- ✓ Serves as an informal review process
  - each line of code is looked at by more than 1 person.
- ✓ It encourages refactoring as the whole team can benefit from improving the system code.
- ✓ Pairs are created dynamically so that all team members work with each other during the development process.
- ✓ Sharing of knowledge during pair programming
  - very important as it reduces the overall risks to a project when team members leave.
- ✓ There is some evidence that suggests that a pair working together is more efficient than 2 programmers working separately.

AGILE PROJECT MANAGEMENT

AGILE PROJECT MANAGEMENT

- ✓ Software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project.

- ✓ Standard approach = project plan
  - what should be delivered,
  - when it should be delivered
  - and who will work on the development of the project deliverables.

- ✓ Agile project management?

SCRUM

- Scrum is an agile method that focuses on managing iterative development rather than specific agile practices.
- There are three phases in Scrum.
  - The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
  - This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
  - The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

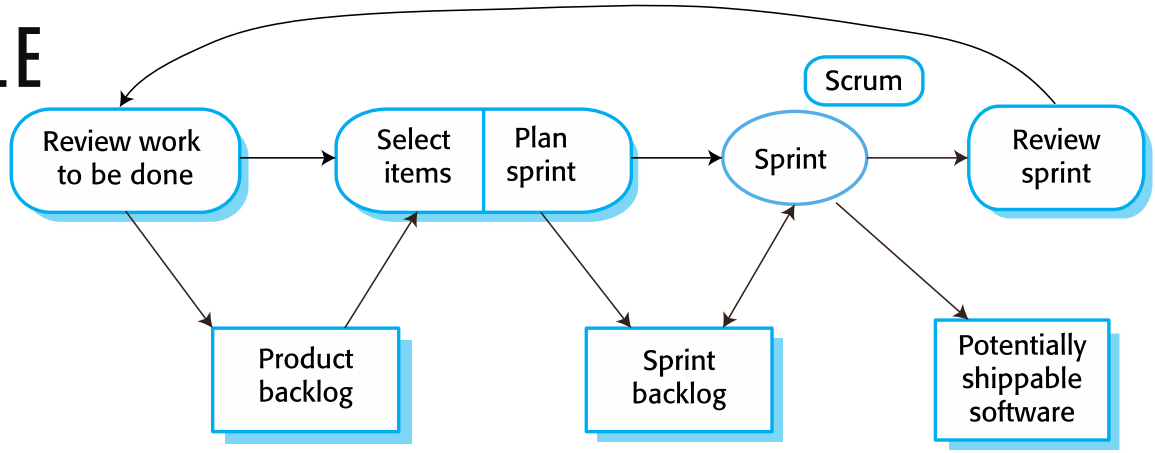
SCRUM TERMINOLOGY (A)

Scrum term	Definition
Development team	A self-organizing group of software developers, which should be no more than 7 people. They are responsible for developing the software and other essential project documents.
Potentially shippable product increment	The software increment that is delivered from a sprint. The idea is that this should be 'potentially shippable' which means that it is in a finished state and no further work, such as testing, is needed to incorporate it into the final product. In practice, this is not always achievable.
Product backlog	This is a list of 'to do' items which the Scrum team must tackle. They may be feature definitions for the software, software requirements, user stories or descriptions of supplementary tasks that are needed, such as architecture definition or user documentation.
Product owner	An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development and continuously review the product backlog to ensure that the project continues to meet critical business needs. The Product Owner can be a customer but might also be a product manager in a software company or other stakeholder representative.

SCRUM TERMINOLOGY (B)

Scrum term	Definition
Scrum	A daily meeting of the Scrum team that reviews progress and prioritizes work to be done that day. Ideally, this should be a short face-to-face meeting that includes the whole team.
ScrumMaster	The ScrumMaster is responsible for ensuring that the Scrum process is followed and guides the team in the effective use of Scrum. He or she is responsible for interfacing with the rest of the company and for ensuring that the Scrum team is not diverted by outside interference. The Scrum developers are adamant that the ScrumMaster should not be thought of as a project manager. Others, however, may not always find it easy to see the difference.
Sprint	A development iteration. Sprints are usually 2-4 weeks long.
Velocity	An estimate of how much product backlog effort that a team can cover in a single sprint. Understanding a team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring improving performance.

SCRUM SPRINT CYCLE

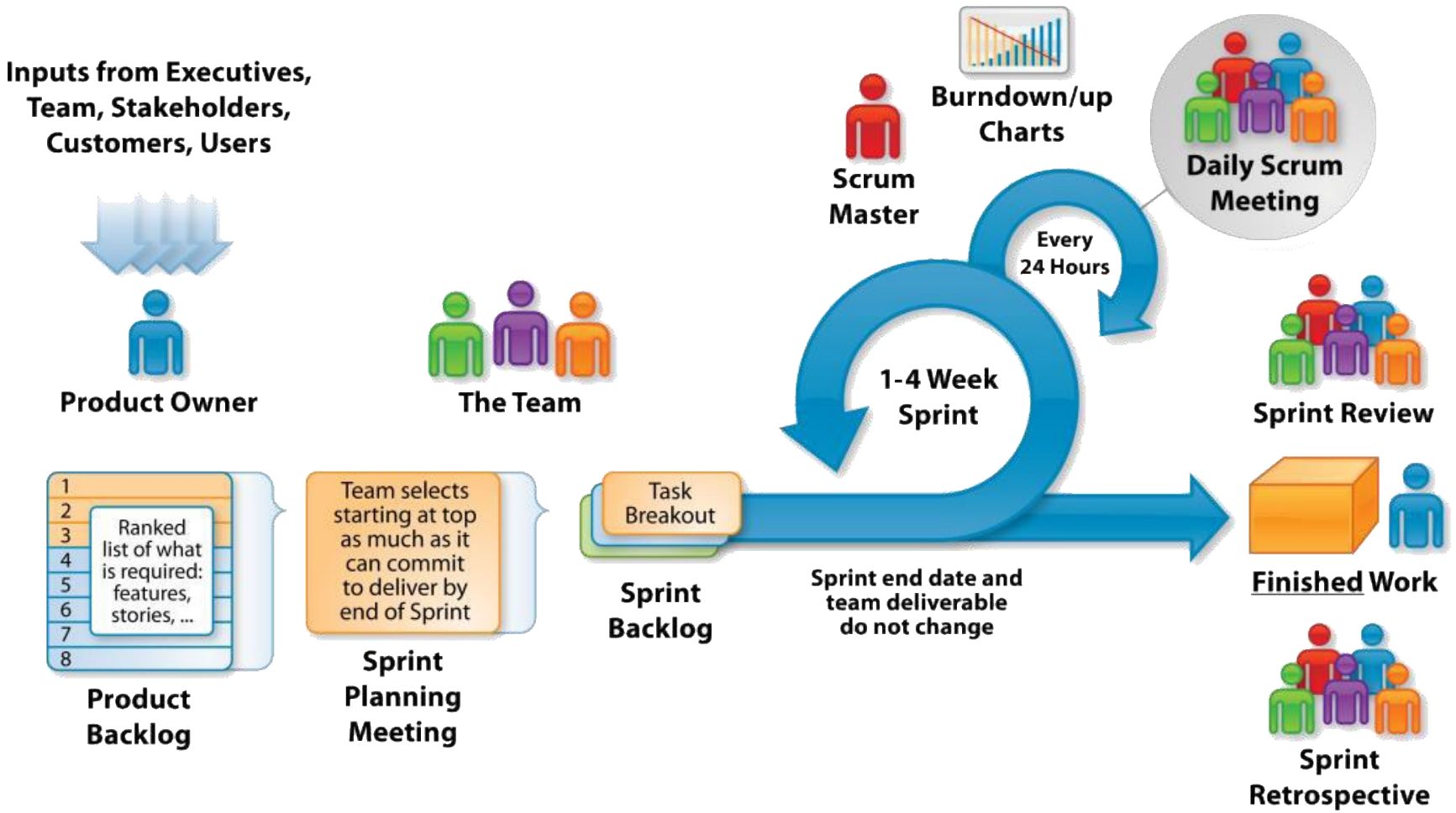


- Sprint: fixed length, normally 2–4 weeks
- Product backlog: list of work to be done on the project

- ✓ The selection: select the features and functionality from the product backlog to be developed during the sprint.
  - The team organize themselves to develop the software.
- ✓ During "sprint" stage:
  - the team is isolated from the customer and the organization
  - all communications through the 'Scrum master'.
    - to protect the development team from external distractions.
- ✓ End of the sprint:
  - the work is reviewed and presented to stakeholders.

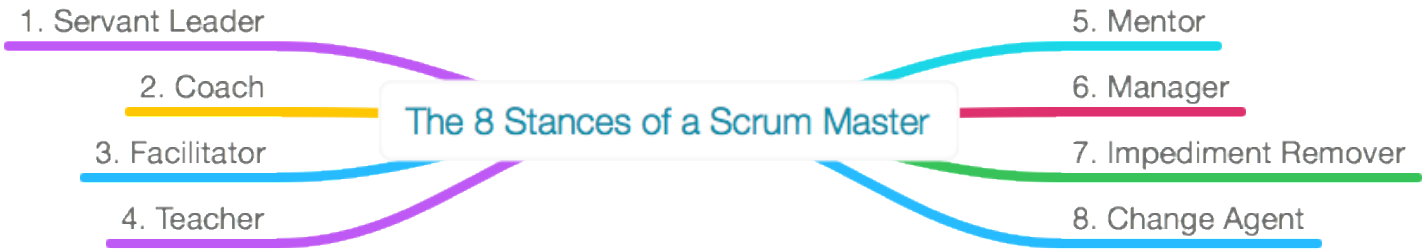
SCRUM SPRINT CYCLE – ANOTHER VIEW

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



TEAMWORK IN SCRUM

- ✓ The 'Scrum master': a facilitator
  - arranges daily meetings & tracks the backlog of work to be done,
  - records decisions & measures progress against the backlog
  - communicates with customers / management outside of the team.



- ✓ Short daily meetings (Scrums) for all team members
  - share information,
  - describe progress since the last meeting,
  - problems that have arisen
  - what is planned for the following day => re-plan if necessary.

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.

SUMMARY

- ✓ Agile methods are incremental development methods that focus on
  - rapid software development,
  - frequent releases of the software,
  - reducing process overheads by minimizing documentation
  - and producing high-quality code.
- ✓ Agile development practices include
  - User stories for system specification
  - Frequent releases of the software,
  - Continuous software improvement
  - Test-first development
  - Customer participation in the development team.

SUMMARY (CONT.)

- ✓ Scrum is an agile method that provides a project management framework.
  - It is centred round a set of sprints, which are fixed time periods when a system increment is developed.
- ✓ Many practical development methods are a mixture of plan-based and agile development.