Chapter 3 - Agile Software Development

MI Qing (Lecturer)

Telephone: 15210503242

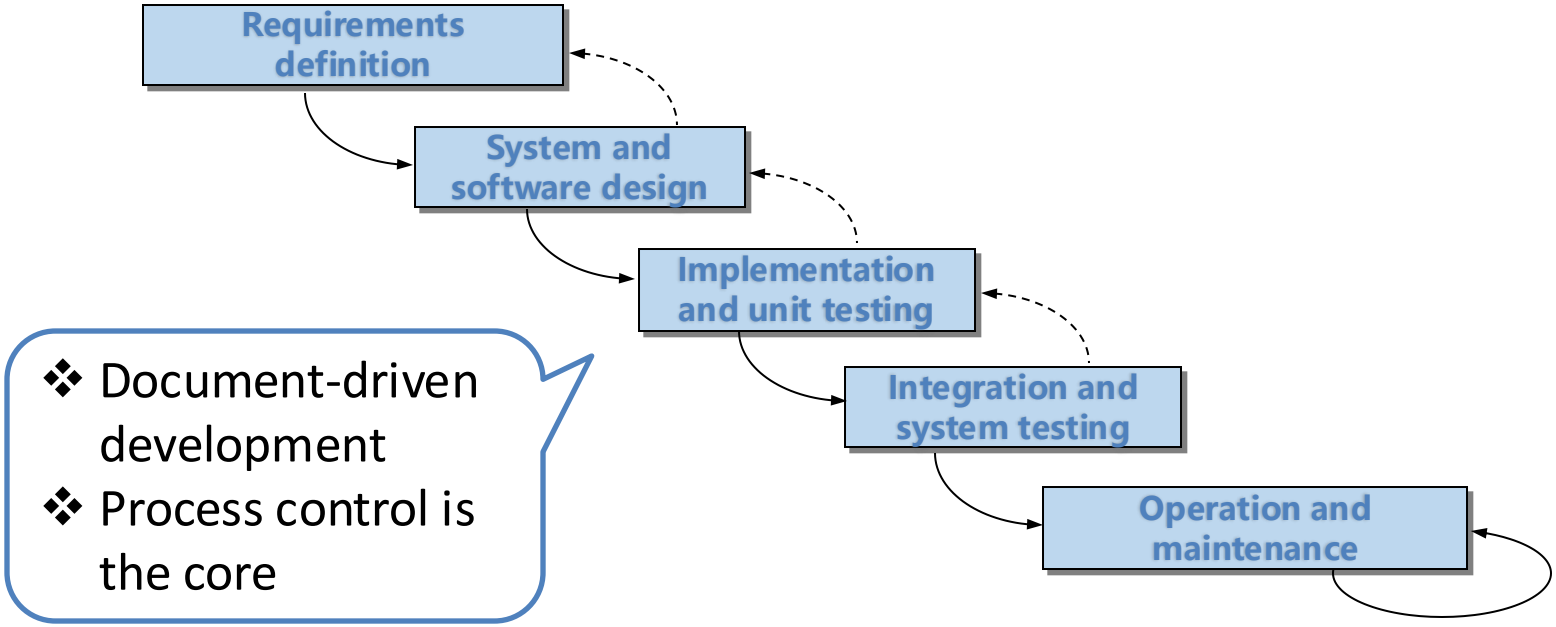
Email: miqing@bjut.edu.cn

Office: 410, Information Building

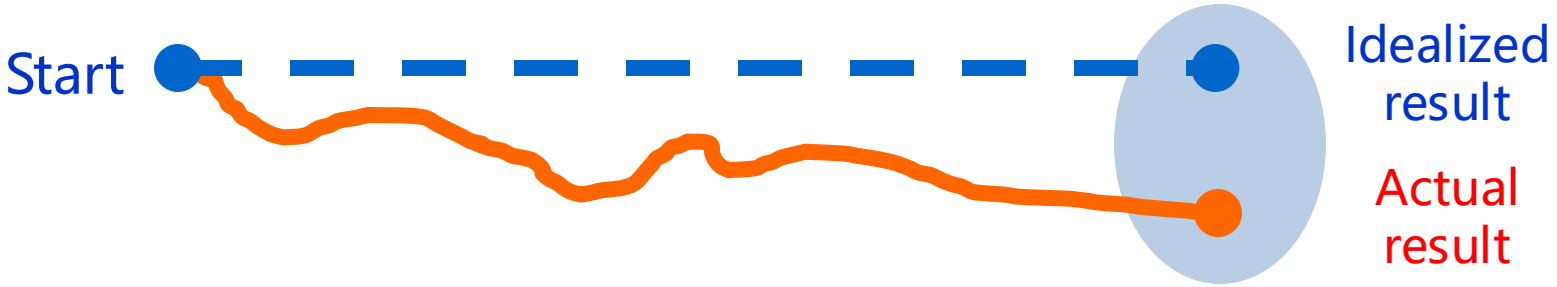
Topics Covered

- ✧ Agile methods
- ✧ Agile development techniques
- ✧ Agile project management
- ✧ Practical problems with agile methods

Traditional Software Development



- Document-driven development
- Process control is the core

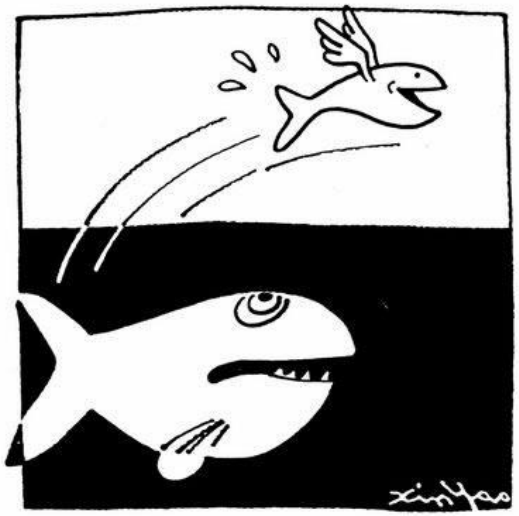


Quality Software

Are we building the right product?

&

Are we building the product right?



- Software development should focus more on delivery
  - High quality deliverables are of great importance
  - Systems are not built at once, but evolve iteratively

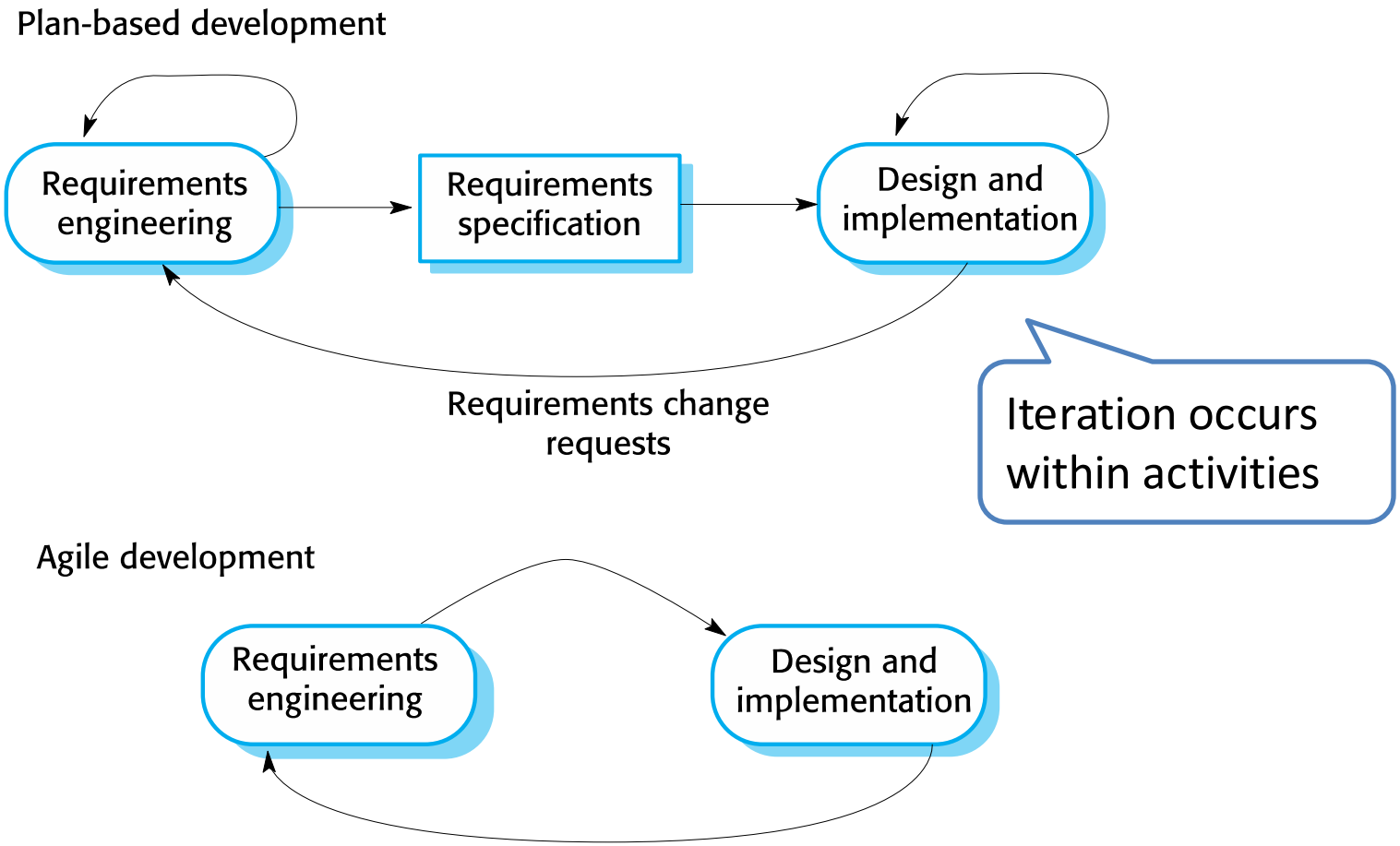
Rapid Software Development

- Rapid development and delivery is now often the most important requirement for software systems.
  - Businesses operate in a fast-changing requirement and it is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs
- Plan-driven development is essential for some types of system but does not meet these business needs.
- Agile development methods emerged in the late 1990s whose aim was to radically reduce the delivery time for working software systems.

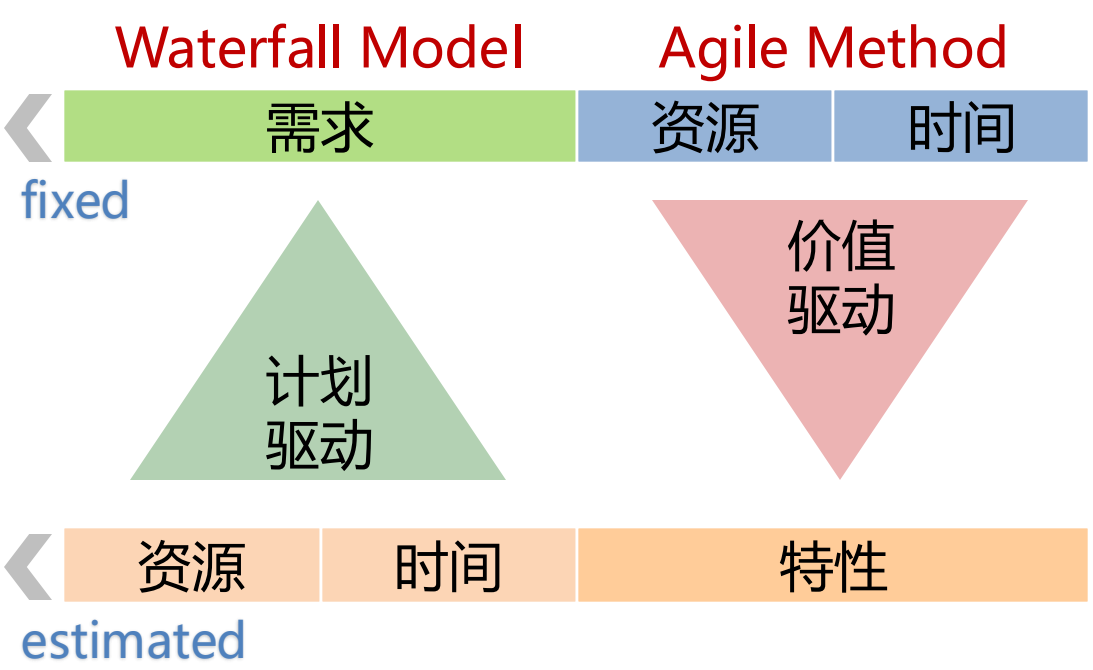
Agile Development

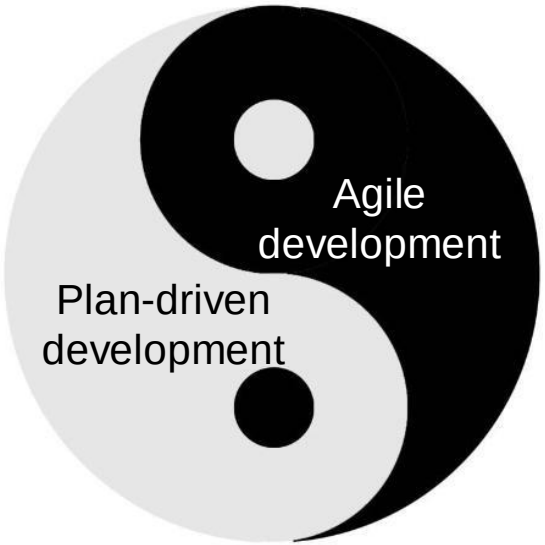
- Program specification, design and implementation are inter-leaved
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g., automated testing tools) used to support development
- Minimal documentation: focus on working code

Plan-driven and Agile Development



The Way of Software Development





动极而静，静极复动
一动一静，互为其根

Agile Methods

Agile Methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These 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
- The aim of agile methods is to reduce overheads in the software process (e.g., by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

Agile Manifesto (https://agilemanifesto.org/)

敏捷宣言

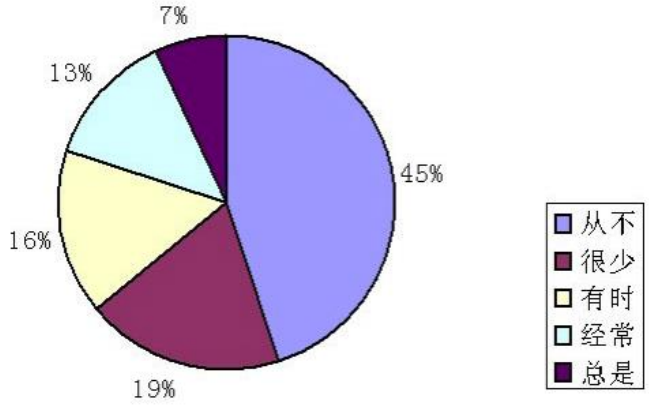
- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - 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.

The Principles of Agile Methods

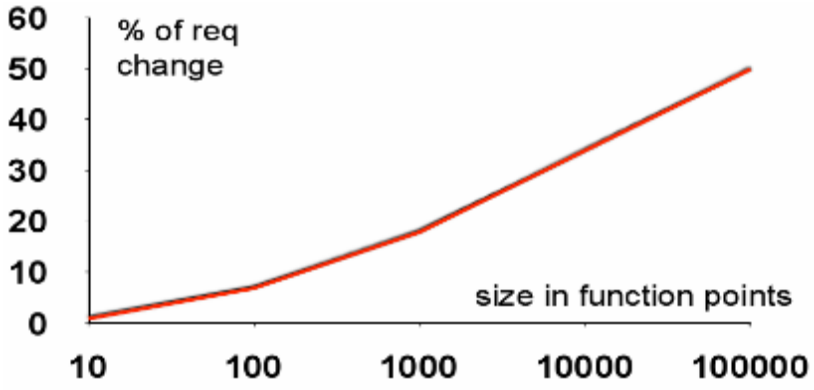
Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer 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.

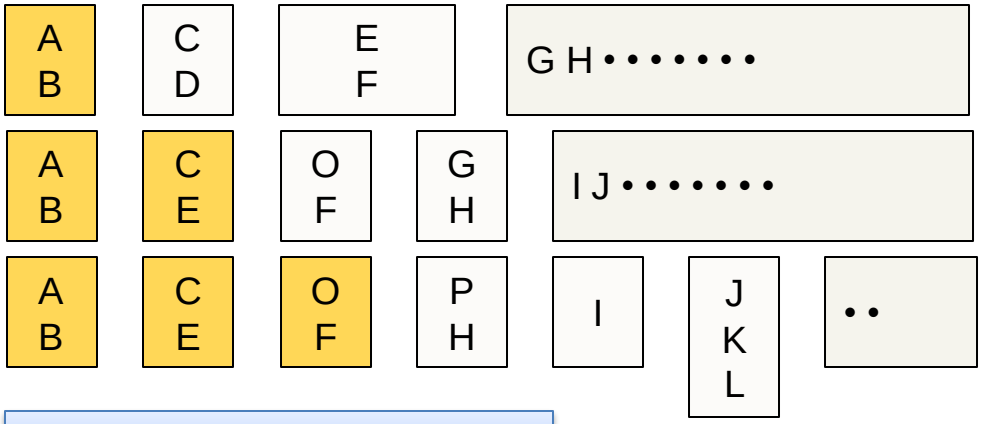
The Principles of Agile Methods

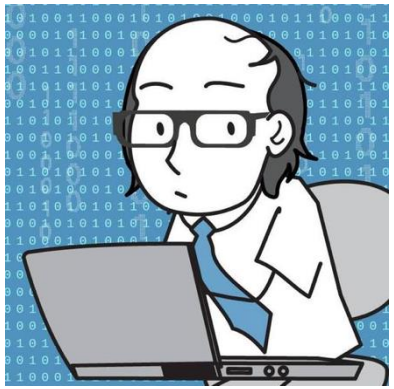
聚焦客户价值



随软件规模增长，需求变化呈非线性增长







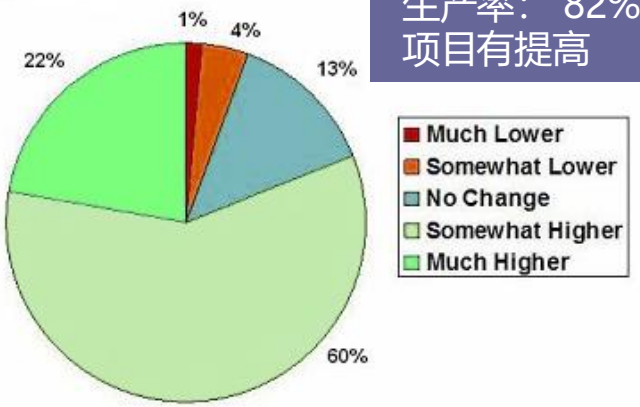
不断调整以适应变化

Agile Method Applicability 敏捷方法适应情况

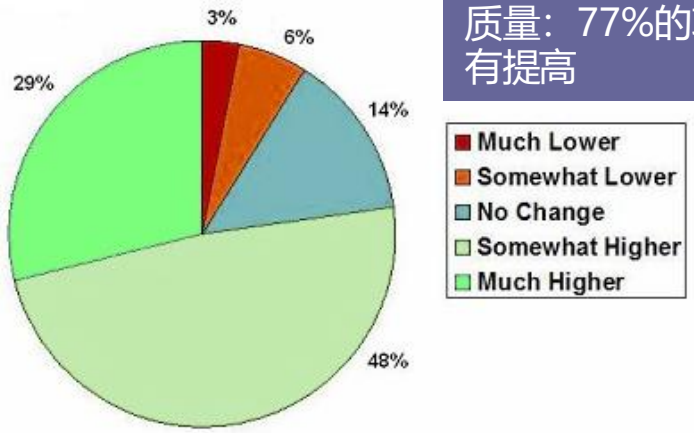
- Product development where a software company is developing a small or medium-sized product for sale.
  - Virtually all software products and apps are now developed using an agile approach
- Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are few external rules and regulations that affect the software.

Effectiveness of Agile Methods

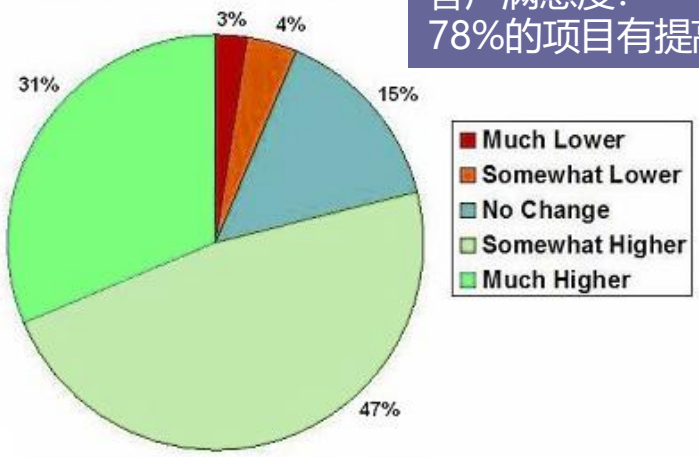
生产率： 82%的项目有提高



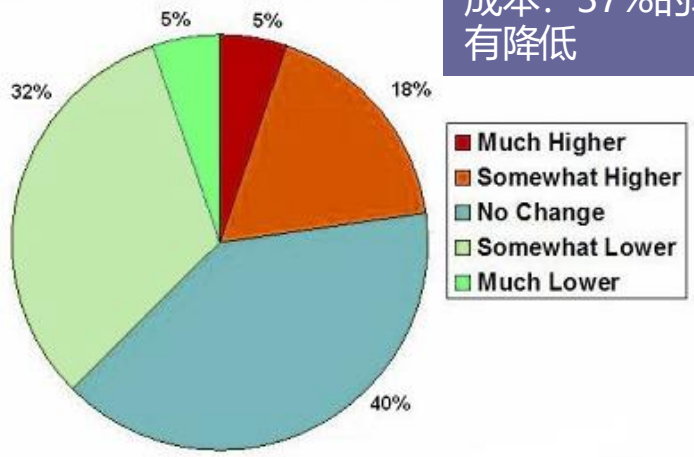
质量：77%的项目有提高



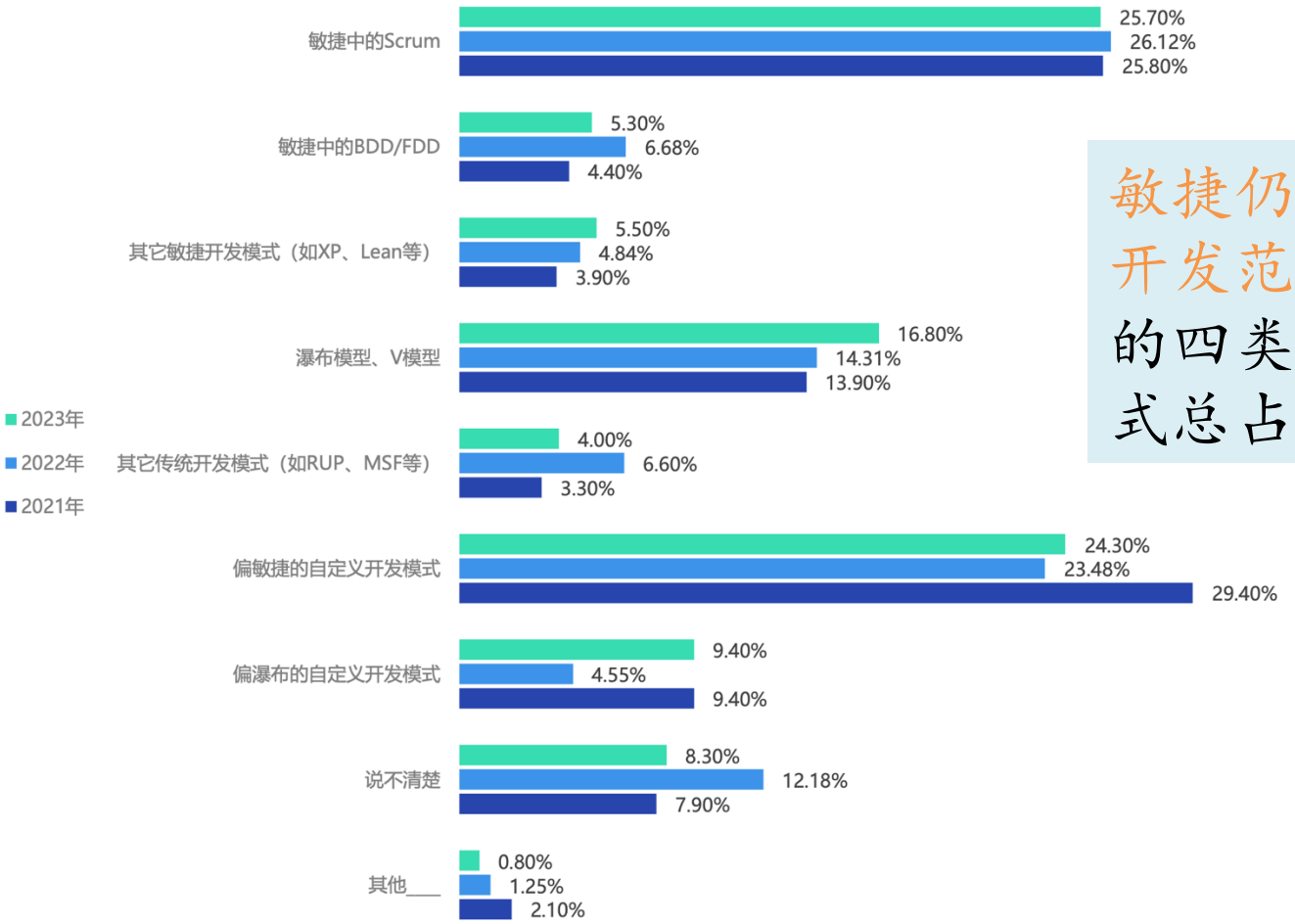
客户满意度：78%的项目有提高



成本：37%的项目有降低



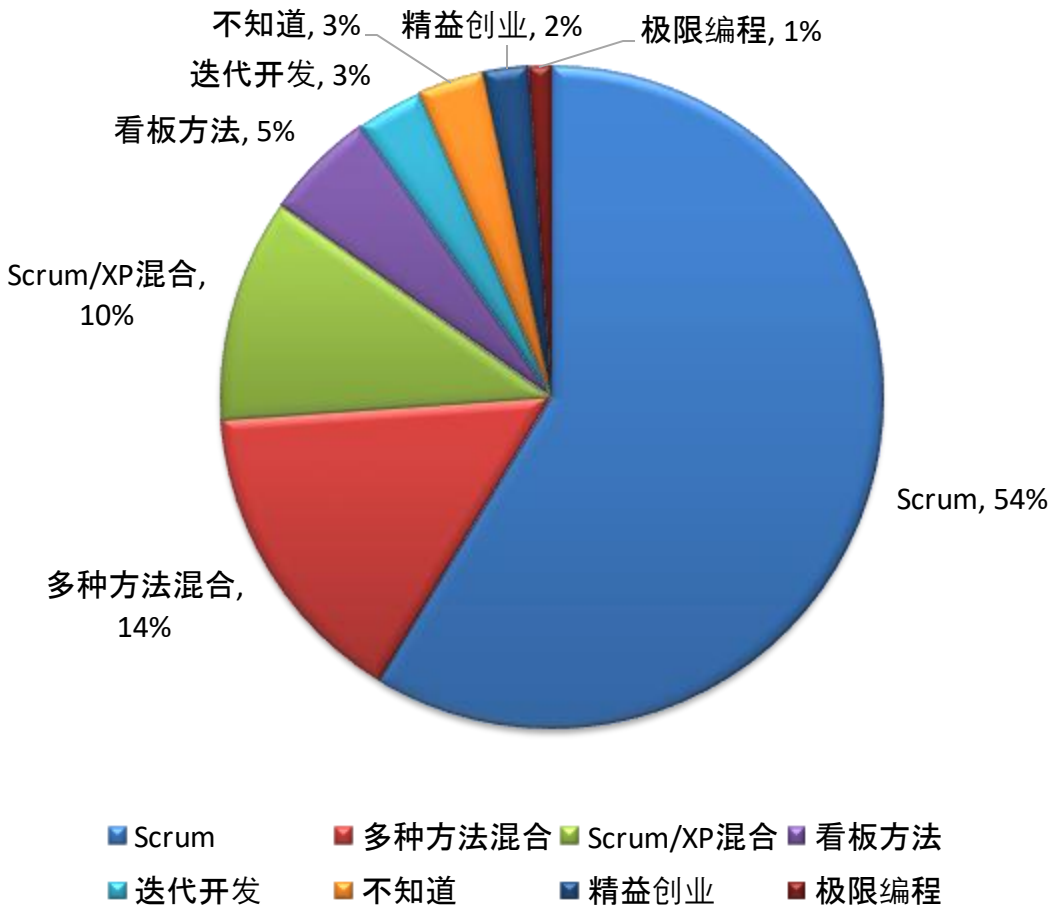
Agile Adoption Rates



敏捷仍然是主流的开发范式，问卷中的四类敏捷开发范式总占比达到60.8%。

数据来源：2023年国内软件质量调查报告

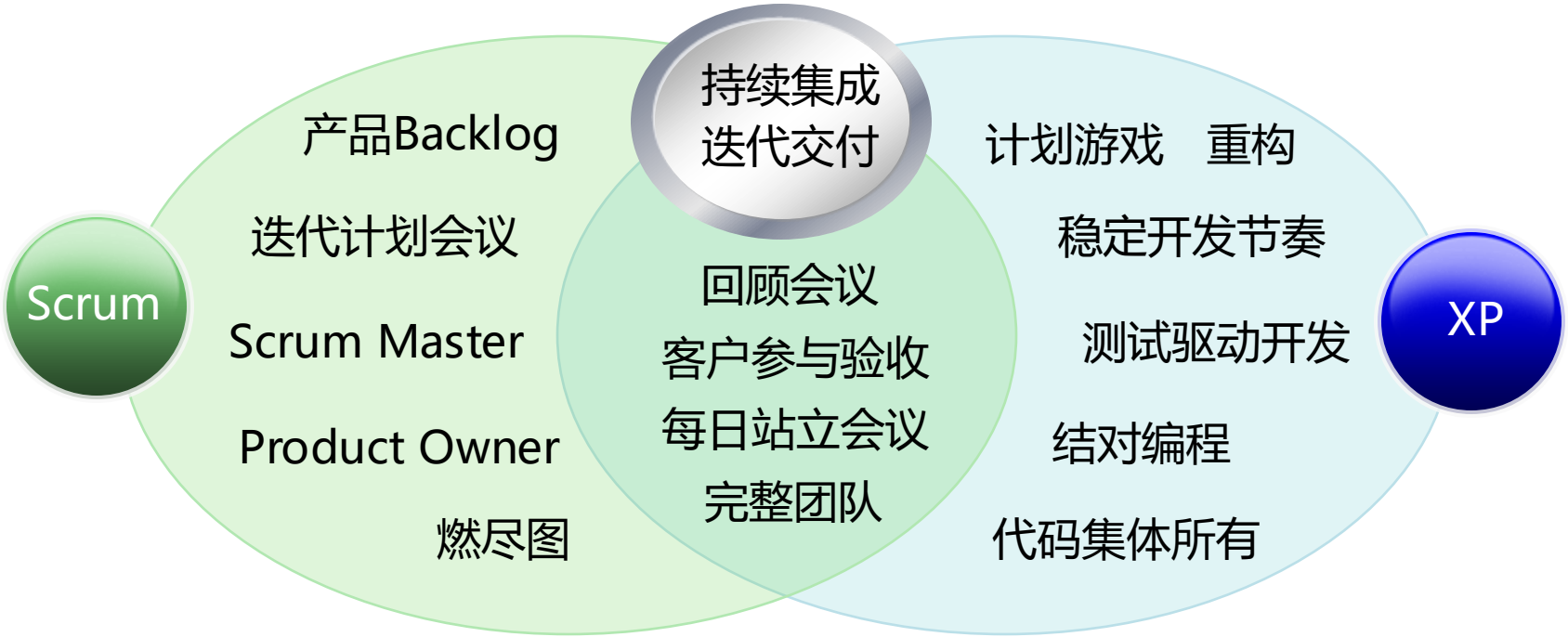
Different Agile Methods



根据2019年的第13届VersionOne版年度敏捷行业状态报告，以Scrum为基础的方法论体系（包括**Scrum**、**Scrum/XP混合**等）仍然居于主流地位，使用率最高。

除此之外，还有**DSDM**、**FDD**、**RAD**等一些符合敏捷精神的小众方法论。

Scrum VS XP

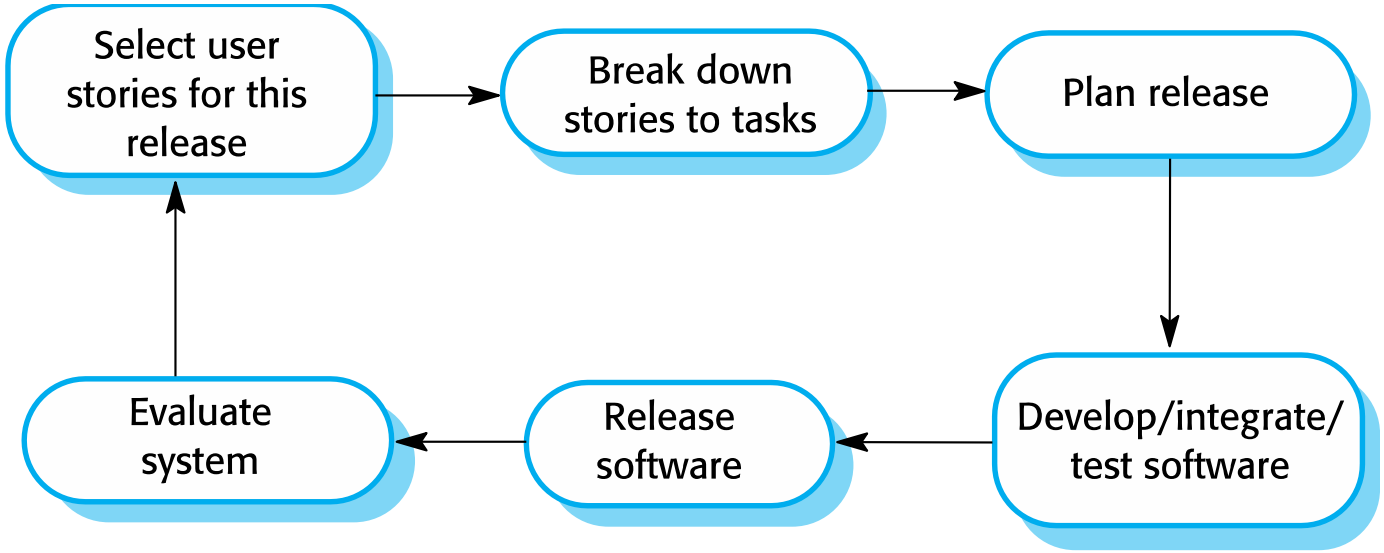


Scrum偏重项目管理　　XP偏重编程实践

Agile Development Techniques

Extreme Programming 极限编程

- A very influential agile method, proposed by Kent Beck in the late 1990s, that introduced a range of agile development techniques.
- Extreme Programming (XP) takes an 'extreme' approach to iterative development.



Extreme Programming Practices (a)

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.

Extreme Programming Practices (b)

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.

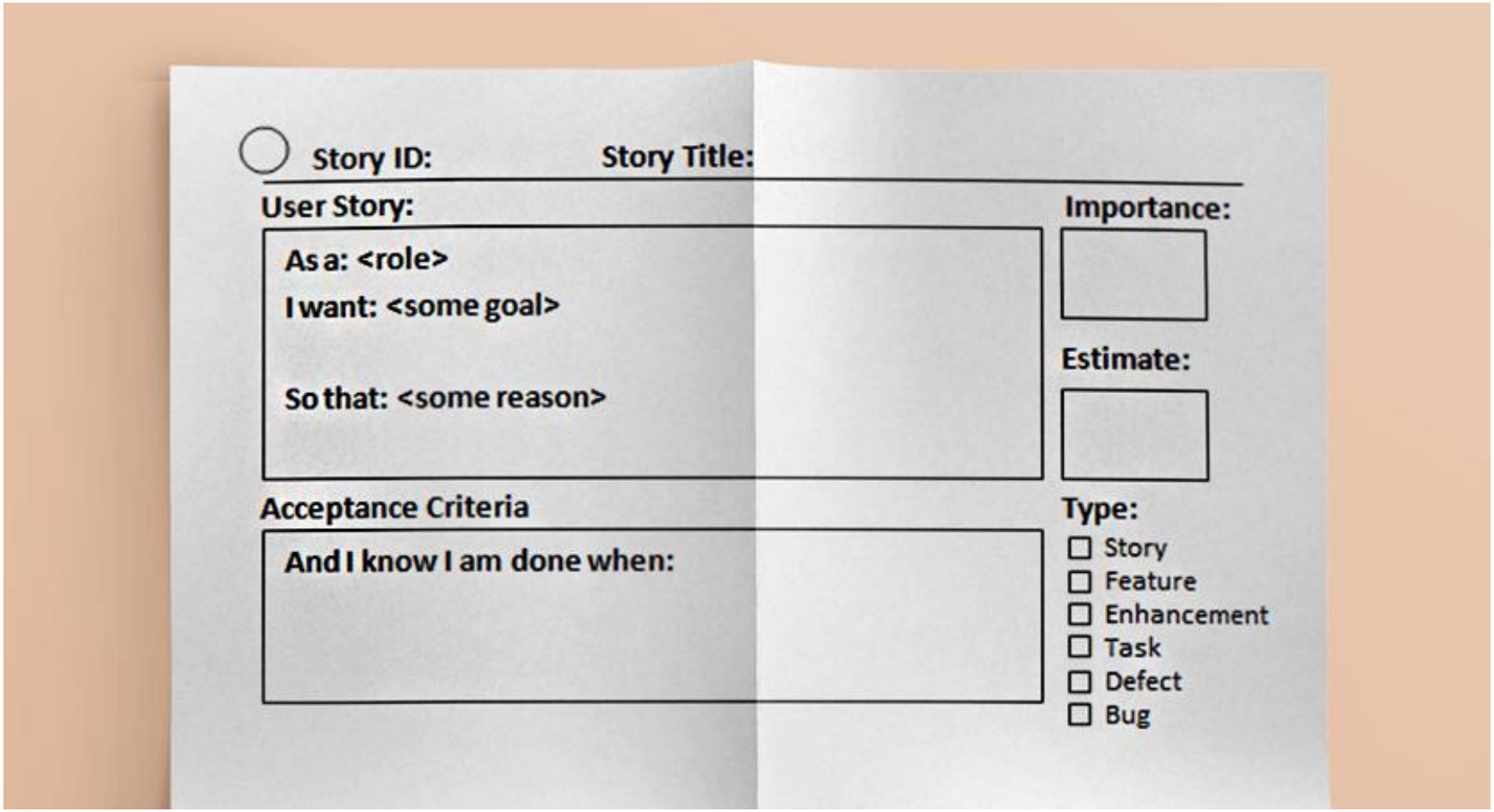
Key Practices

- ✧ 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.

User Story Card



User Story Examples

优先级	名称	用户故事描述
1	浏览商品	作为一名顾客想购买商品而不确定型号时，我希望能浏览网站在售的商品，按照①商品类型和②价格范围进行过滤。
2	搜索商品	作为一名顾客在查找某种商品时，我希望能进行不限格式的文本搜索，例如按照短语或关键字。
3	注册账户	作为一名新顾客，我希望注册并设置一个帐户，包括用户名、密码、信用卡和送货信息等。
4	维护购物车	作为一名顾客，我希望能将指定商品放入购物车（稍后购买）、查看我的购物车内的商品以及移除我不想要的物品。
5	结账	作为一名顾客，我希望能完成我购物车内所有商品的购买过程。
6	编辑商品规格	作为一名工作人员，我希望能够添加和编辑在售商品的详细信息（包括介绍、规格说明、价格等）。
7	查看订单	作为一名工作人员，我希望能登录并查看一段时间内应该完成或已经完成的所有订单。

Planning Poker

- Planning Poker is an agile estimating and planning technique that is consensus based. To start a poker planning session, the product owner reads an agile user story or describes a feature to the estimators.
- Each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100. The values represent the number of story points, ideal days, or other units in which the team estimates.

Planning Poker

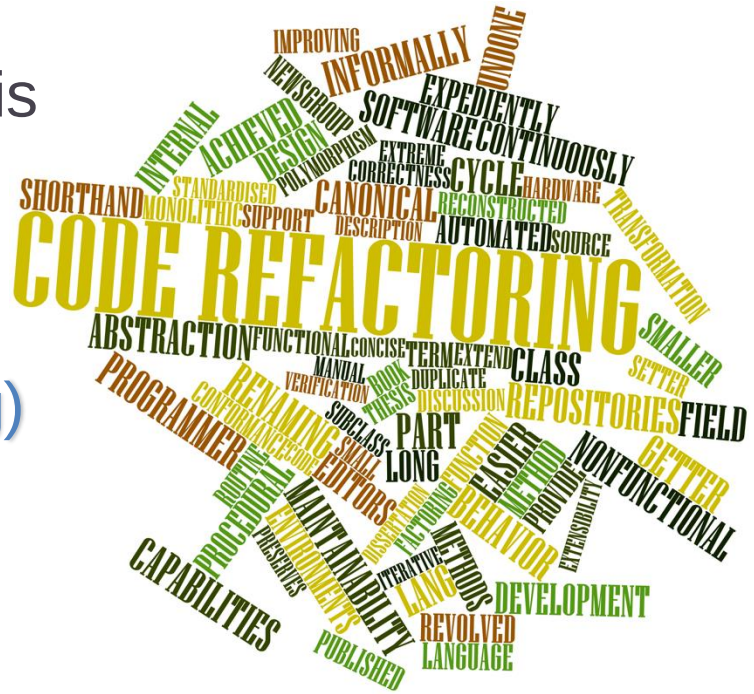
- When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time.
- If all estimators selected the same value, that becomes the estimate. If not, the estimators discuss their estimates. The high and low estimators should especially share their reasons.

Planning Poker

- ✧ After further discussion, each estimator reselects an estimate card, and all cards are again revealed at the same time.
- ✧ The poker planning process is repeated until consensus is achieved or until the estimators decide that agile estimating and planning of a particular item needs to be deferred until additional information can be acquired.

Refactoring

- Conventional wisdom in software engineering is to design for change. It is worth spending time and effort anticipating changes as this reduces costs later in the life cycle.
- XP, however, maintains that this is not worthwhile as changes cannot be reliably anticipated.
- Rather, it proposes constant code improvement (refactoring) to make changes easier when they have to be implemented.

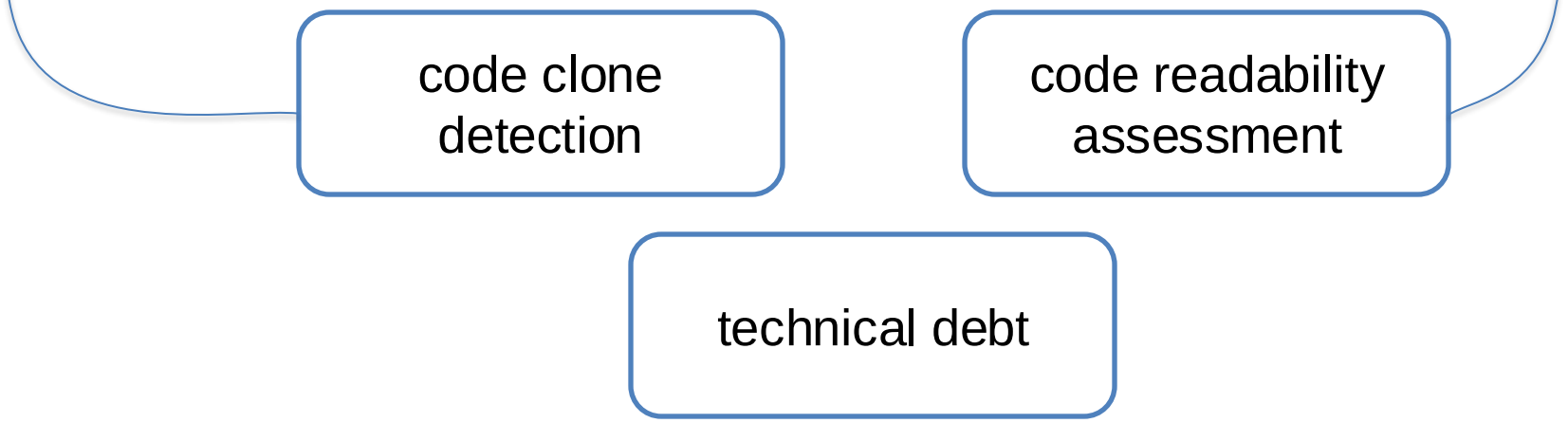


Refactoring

- Programming team look for possible software improvements and make these improvements even where there is no immediate need for them.
- This improves the understandability of the software and so reduces the need for documentation.
- Changes are easier to make because the code is well-structured and clear.
- However, some changes requires architecture refactoring and this is much more 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.

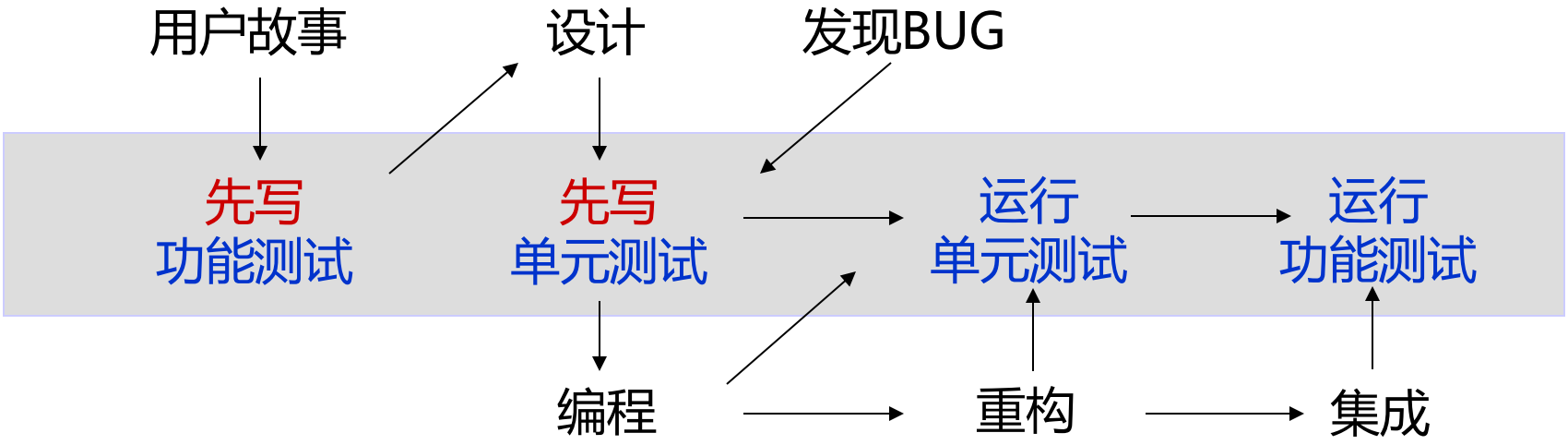


Test-first Development

- 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.
- Writing tests before code clarifies the requirements to be implemented.

Test-driven Development

- 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.
- All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.



Test Automation

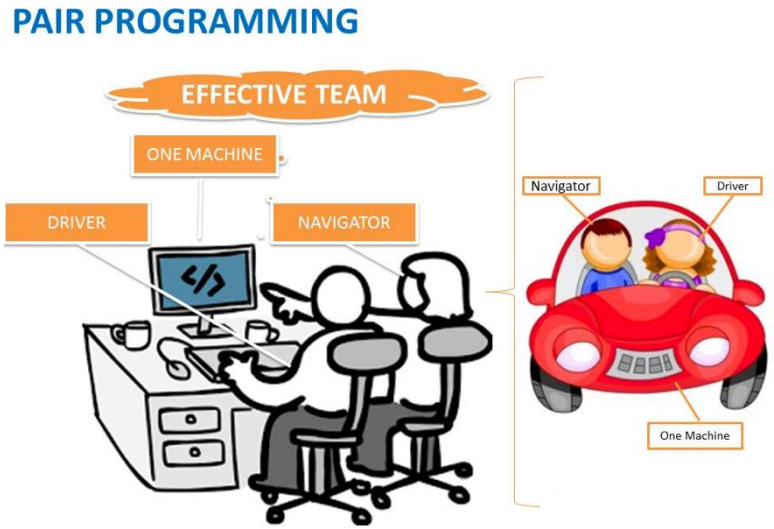
- Test automation means that tests are written as executable components before the task is implemented.
  - These testing components should be stand-alone, should simulate the submission of input to be tested and should check that the result meets the output specification. An automated test framework (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 is 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 involves programmers working in pairs, developing code together.
- Benefits:
  - This helps develop common ownership of code and spreads knowledge across the team.
  - It serves as an informal review process as 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.



Pair Programming

Mark

- In pair programming, programmers sit together at the same computer to develop the software.
- Pairs are created dynamically so that all team members work with each other during the development process.
- The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave.
- Pair programming is not necessarily inefficient and there is some evidence that suggests that a pair working together is more efficient than 2 programmers working separately.

XP and Agile Principles

- Incremental development is 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.

Agile Project Management

Creators of Scrum

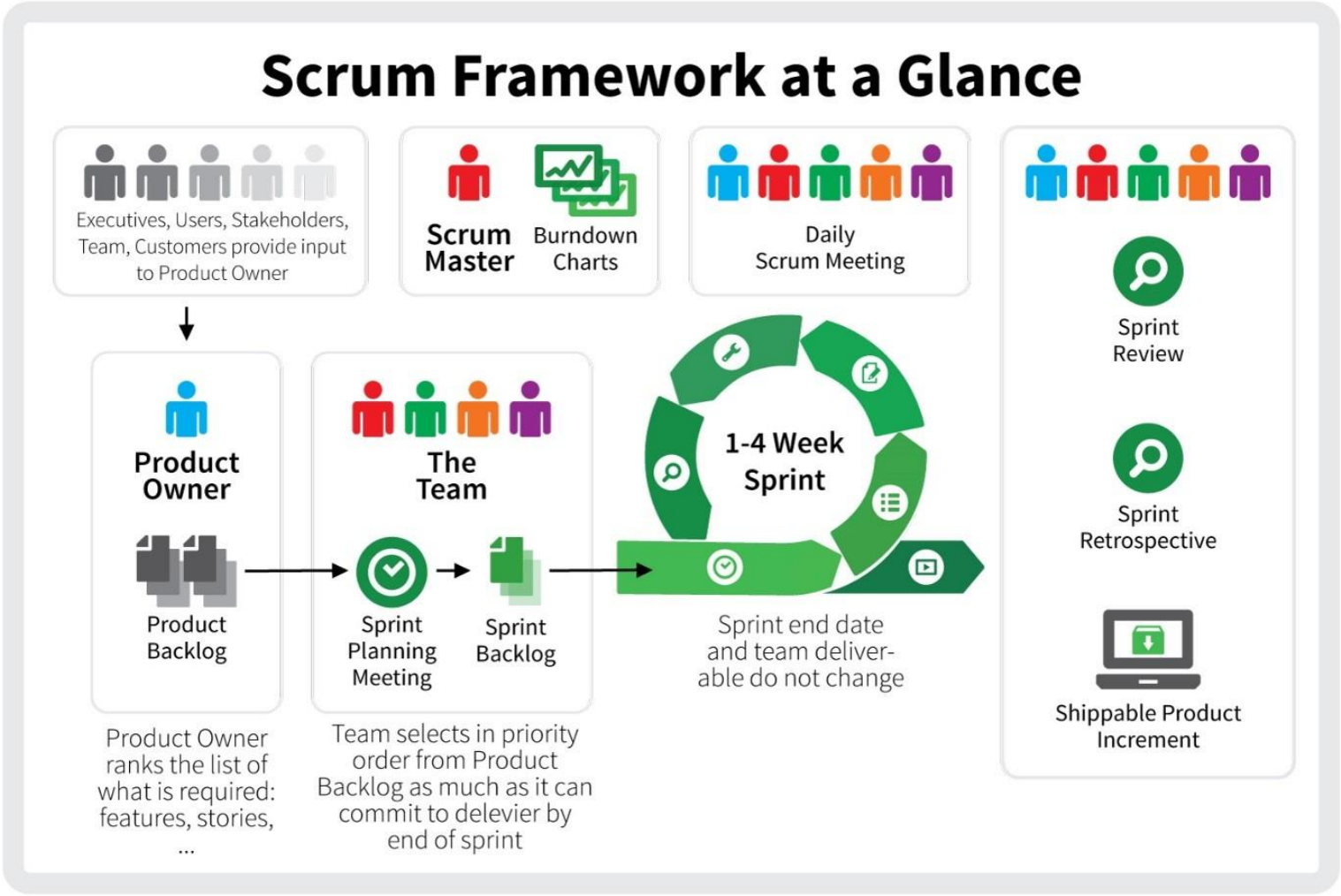
The scrum process has its origins in the early 1990s. Jeff Sutherland and Ken Schwaber come up with process, which they presented to the Object-Oriented Programming, Systems, Languages & Applications (OOPSLA) conference in Austin, Texas in 1995.

Definition of Scrum

- Scrum: a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
- Scrum is:
  - Lightweight
  - Simple to understand
  - Difficult to master
- Scrum is not a process, technique, or definitive method. Rather, it is a framework within which you can employ various processes and techniques.

Scrum Framework

Scrum中的三种角色



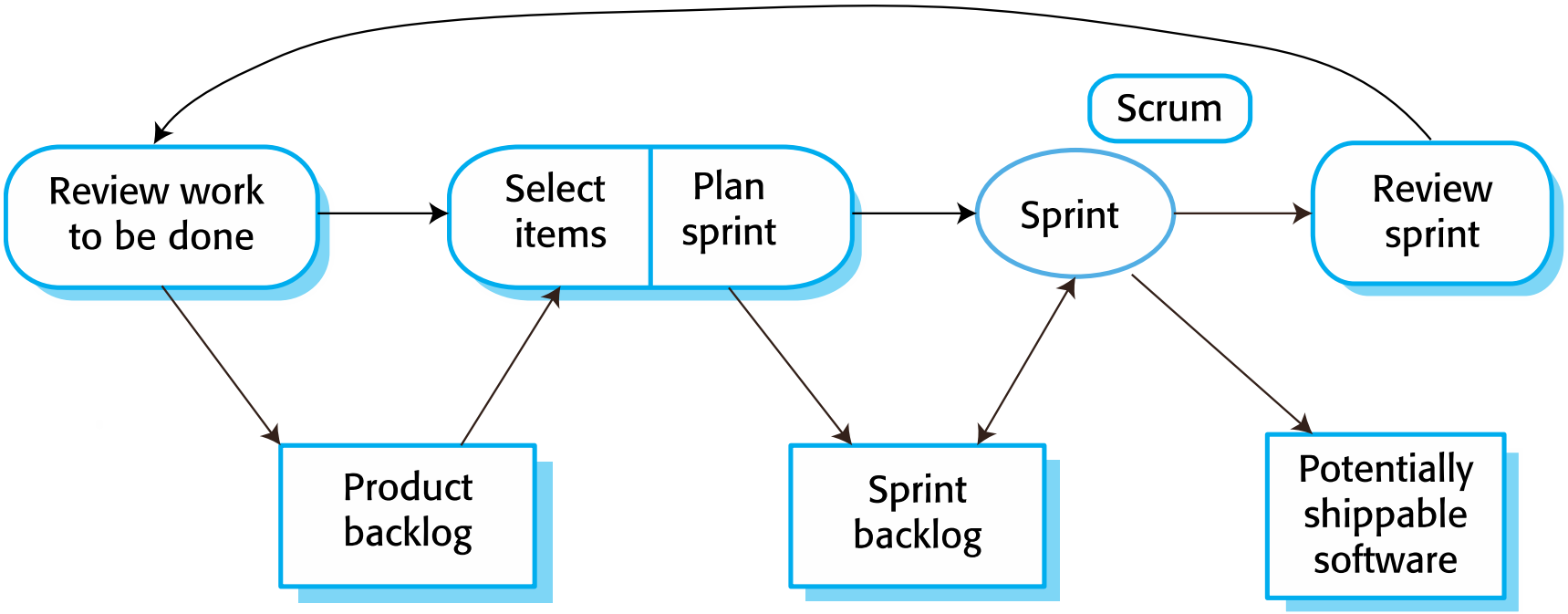
Scrum Terminology (a)

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.
Scrum Master	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.

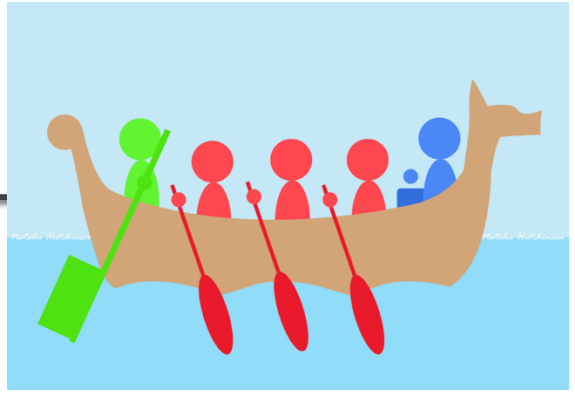
Scrum Sprint Cycle



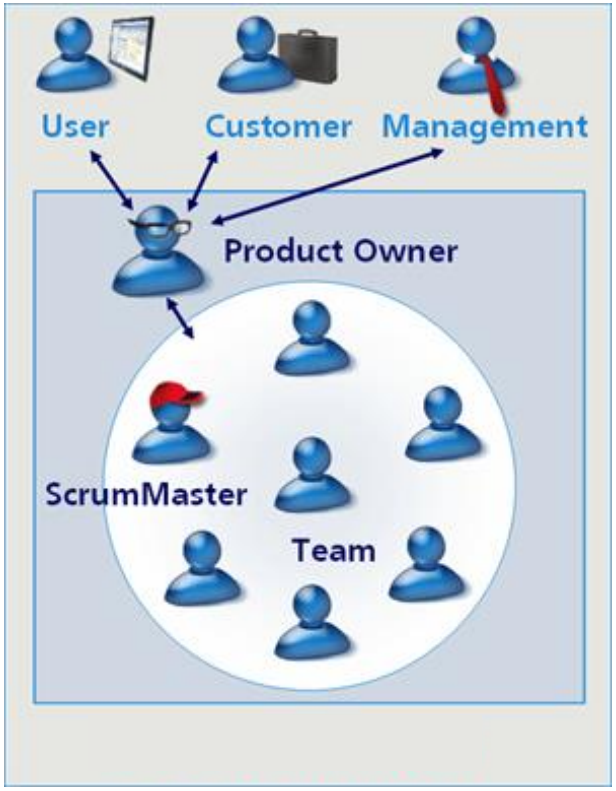
Scrum Sprint Cycle

- Sprints are fixed length, normally 2-4 weeks.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase involves all of the project team to select the features and functionality from the product backlog to be developed during the sprint.
- Once these are agreed, the team organize themselves to develop the software.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

Roles in Scrum



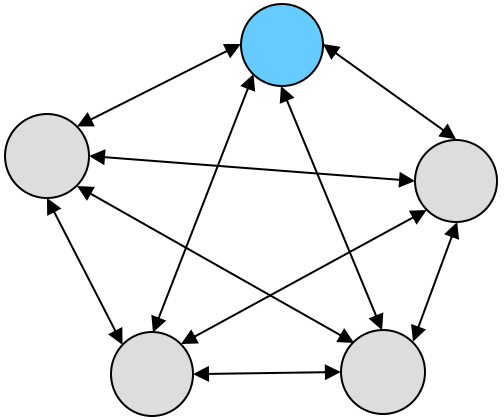
- **Product Owner (green figure)**
  - The “What”: with a focus on value, time to market, return on investment.
- **Development Team (red figures)**
  - The "How”: focus on building something that is useable and potentially releasable.
- **Scrum Master (blue figure)**
  - A facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog.



Scrum Team

✧ Scrum teams are self-organizing and cross-functional

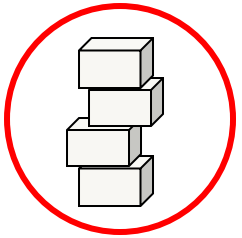
- Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team.
- Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team.



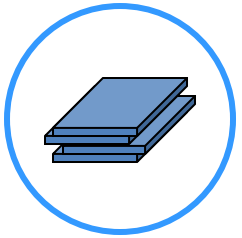
Scrum Artifacts

了解

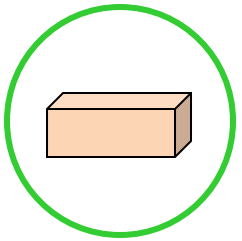
- The Product Backlog is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.
- The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal. The Sprint Backlog is a forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a “Done” Increment.



Product Backlog



Sprint Backlog



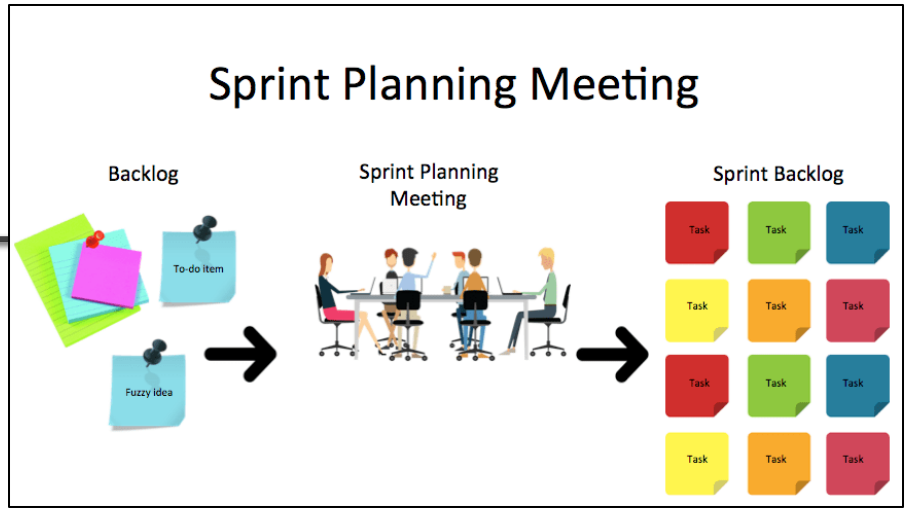
Working Software

Scrum Theory

四种会议

- Scrum is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known.
- Three pillars uphold every implementation of empirical process control: transparency, inspection, and adaptation.
- Scrum prescribes four formal events for inspection and adaptation: Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective.

Sprint Planning



- The work to be performed in the Sprint is planned at the Sprint Planning. This plan is created by the collaborative work of the entire Scrum Team.
- Sprint Planning answers the following:
  - Topic One: What can be done this Sprint?
  - Topic Two: How will the chosen work get done?
- The input to this meeting is the Product Backlog, the latest product Increment, projected capacity of the Development Team during the Sprint, and past performance of the Development Team.

Daily Scrum

- The Daily Scrum is a 15-minute time-boxed event for the Development Team. At it, the Development Team plans work for the next 24 hours.
  - What did I do yesterday that helped the Development Team meet the Sprint Goal?
  - What will I do today to help the Development Team meet the Sprint Goal?
  - Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

- This means that everyone on the team knows what is going on and, if problems arise, can re-plan work to cope with them. This is a key inspect and adapt meeting.

Sprint Review

- A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed. This is an informal meeting, not a status meeting, and the presentation of the Increment is intended to elicit feedback and foster collaboration.
- This is at most a four-hour meeting for one-month Sprints.
- The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint.

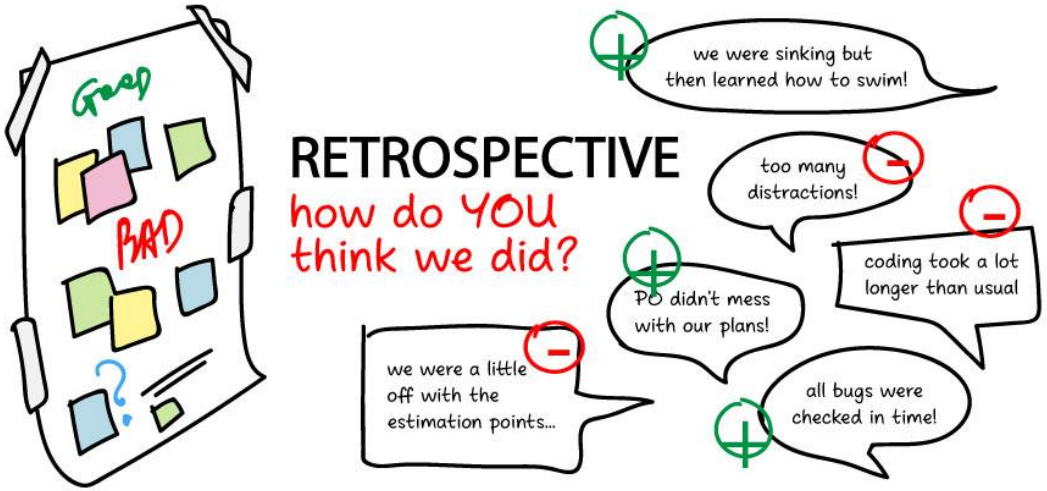
Sprint Review

✧ The Sprint Review includes the following elements:

- Attendees include the Scrum Team and key stakeholders invited by the Product Owner;
- The Product Owner explains what Product Backlog items have been “Done” and what has not been “Done”;
- The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved;
- The Development Team demonstrates the work that it has “Done” and answers questions about the Increment;
- The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint Planning.

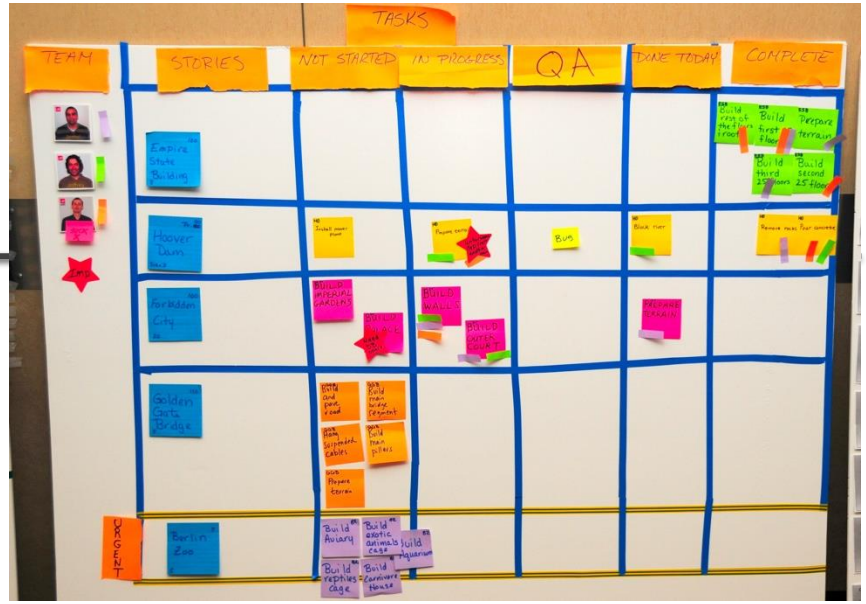
Sprint Retrospective

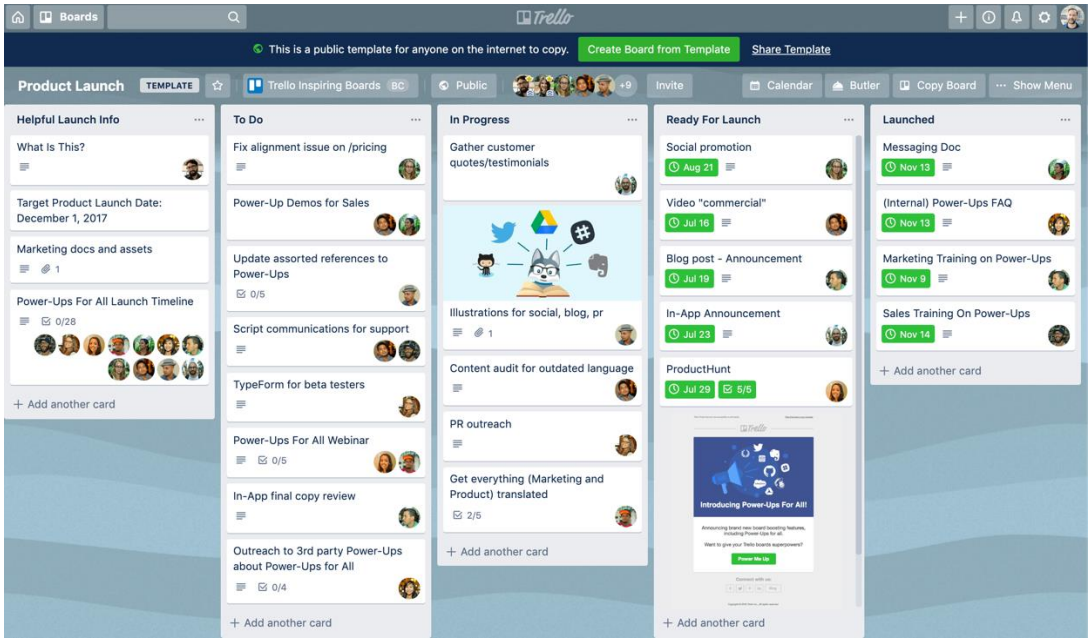
- The Sprint Retrospective is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.
  - The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. This is at most a three-hour meeting for one-month Sprints.
- By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint.



Visual Management

- Scrum Task Board
  - Trello
  - Jira
  - monday.com
  - Asana
  - Basecamp
  - Airtable
  - Freedcamp
  - Scoro

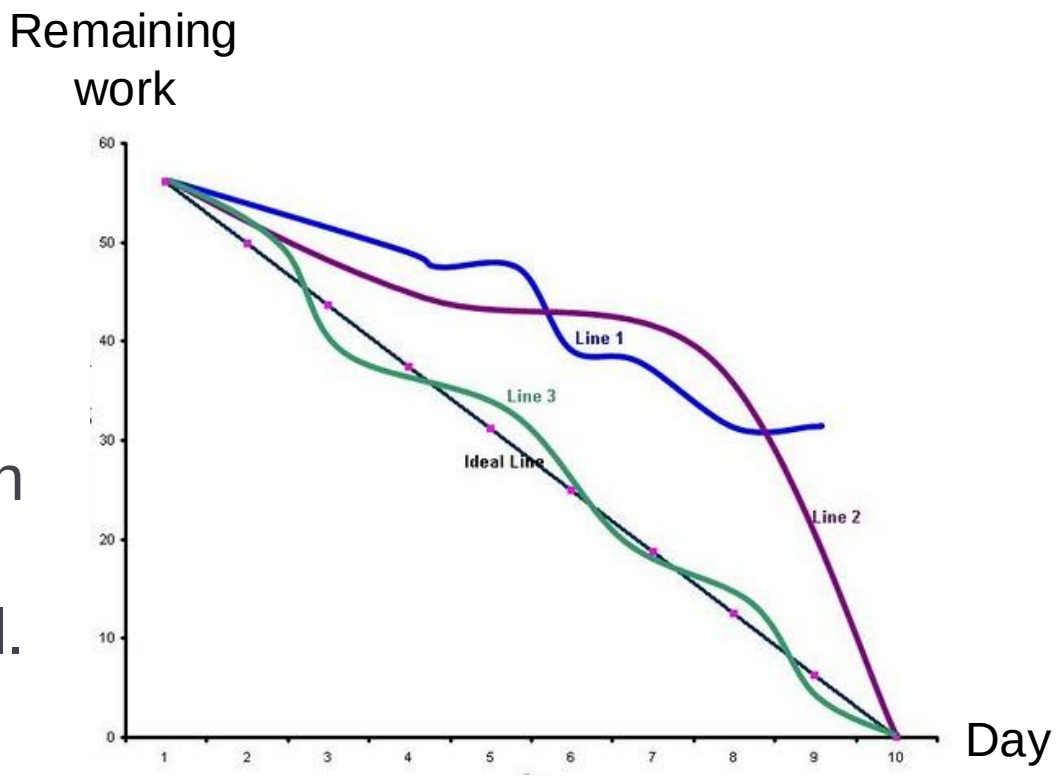




Visual Management

Mark

A burndown chart is a graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal.



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.

Scrum VS XP

区别

Scrum	XP
Typically from two weeks to one month long.	Typically one or two weeks long.
Do not allow changes in their timelines.	Allow changes in their set timelines.
Scrum emphasizes self-organization.	Extreme Programming emphasizes strong engineering practices
In Scrum framework, team determines the sequence in which the product will be developed.	In Extreme Programming, team have to follow a strict priority order or pre-determined priority order.
Scrum framework is not fully described. If you want to adopt it then you need to fill the framework with your own frameworks method like XP or Kanban.	Extreme Programming can be directly applied to a team. XP is also known for its ready-to-apply features.

Practical Problems with Agile Methods

Practical Problems with Agile Methods

- The informality of agile development is incompatible with the legal approach to contract definition that is commonly used in large companies.
- Agile methods are most appropriate for new software development rather than software maintenance. Yet the majority of software costs in large companies come from maintaining their existing software systems.
- Agile methods are designed for small co-located teams yet much software development now involves worldwide distributed teams.

Contractual Issues

- Most software contracts for custom systems are based around a specification, which sets out what has to be implemented by the system developer for the system customer.
- However, this precludes interleaving specification and development as is the norm in agile development.
- A contract that pays for developer time rather than functionality is required.
  - However, this is seen as a high risk by many legal departments because what has to be delivered cannot be guaranteed.

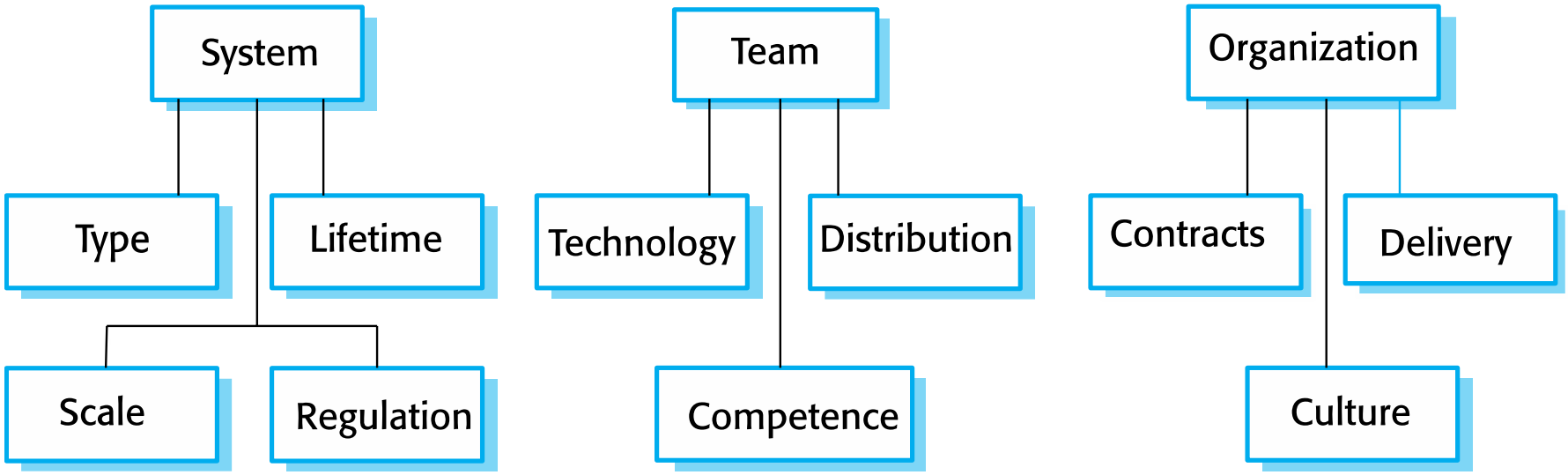
Maintenance Issues

- Key problems are:
  - Lack of product documentation
  - Keeping customers involved in the development process
  - Maintaining the continuity of the development team
- Many agile methods collect requirements informally and incrementally and do not create a coherent requirements document. The use of agile methods may therefore make subsequent system maintenance more difficult.
- Agile development relies on the development team knowing and understanding what has to be done. However, the original developers may not work on the system.

Agile Principles and Practical Problems

Principle	Practice
Customer involvement	This depends on having a customer who is willing and able to spend time with the development team and who can represent all system stakeholders. Often, customer representatives have other demands on their time and cannot play a full part in the software development. Where there are external stakeholders, such as regulators, it is difficult to represent their views to the agile team.
Embrace change	Prioritizing changes can be extremely difficult, especially in systems for which there are many stakeholders. Typically, each stakeholder gives different priorities to different changes.
Incremental delivery	Rapid iterations and short-term planning for development does not always fit in with the longer-term planning cycles of business planning and marketing. Marketing managers may need to know what product features several months in advance to prepare an effective marketing campaign.
Maintain simplicity	Under pressure from delivery schedules, team members may not have time to carry out desirable system simplifications.
People not process	Individual team members may not have suitable personalities for the intense involvement that is typical of agile methods, and therefore may not interact well with other team members.

Agile and Plan-based Factors



System Issues

- How large is the system being developed?
  - Agile methods are most effective a relatively small co-located team who can communicate informally.
- What type of system is being developed?
  - Systems that require a lot of analysis before implementation need a fairly detailed design to carry out this analysis.
- What is the expected system lifetime?
  - Long-lifetime systems require documentation to communicate the intentions of the system developers to the support team.
- Is the system subject to external regulation?
  - If a system is regulated you will probably be required to produce detailed documentation as part of the system safety case.

Team Issues

- How good are the designers and programmers in the development team?
  - It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code.
- How is the development team organized?
  - Design documents may be required if the team is distributed.
- What support technologies are available?
  - IDE support for visualisation and program analysis is essential if design documentation is not available.

Organization Issues

- Traditional engineering organizations have a culture of plan-based development, as this is the norm in engineering.
  - Is it important to have a very detailed specification and design before moving to implementation? If so, you probably need to use a plan-driven approach.
  - Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic? Will customer representatives be available to provide feedback of system increments?
  - Can informal agile development fit into the organizational culture of detailed documentation?

Summary

Key Points

- 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 (XP) include:
  - User stories for specification, Refactoring, Test-first development and Pair programming
- 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.