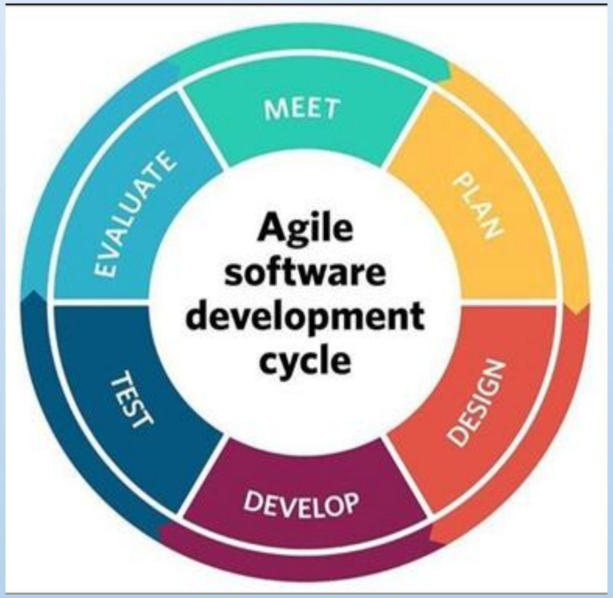
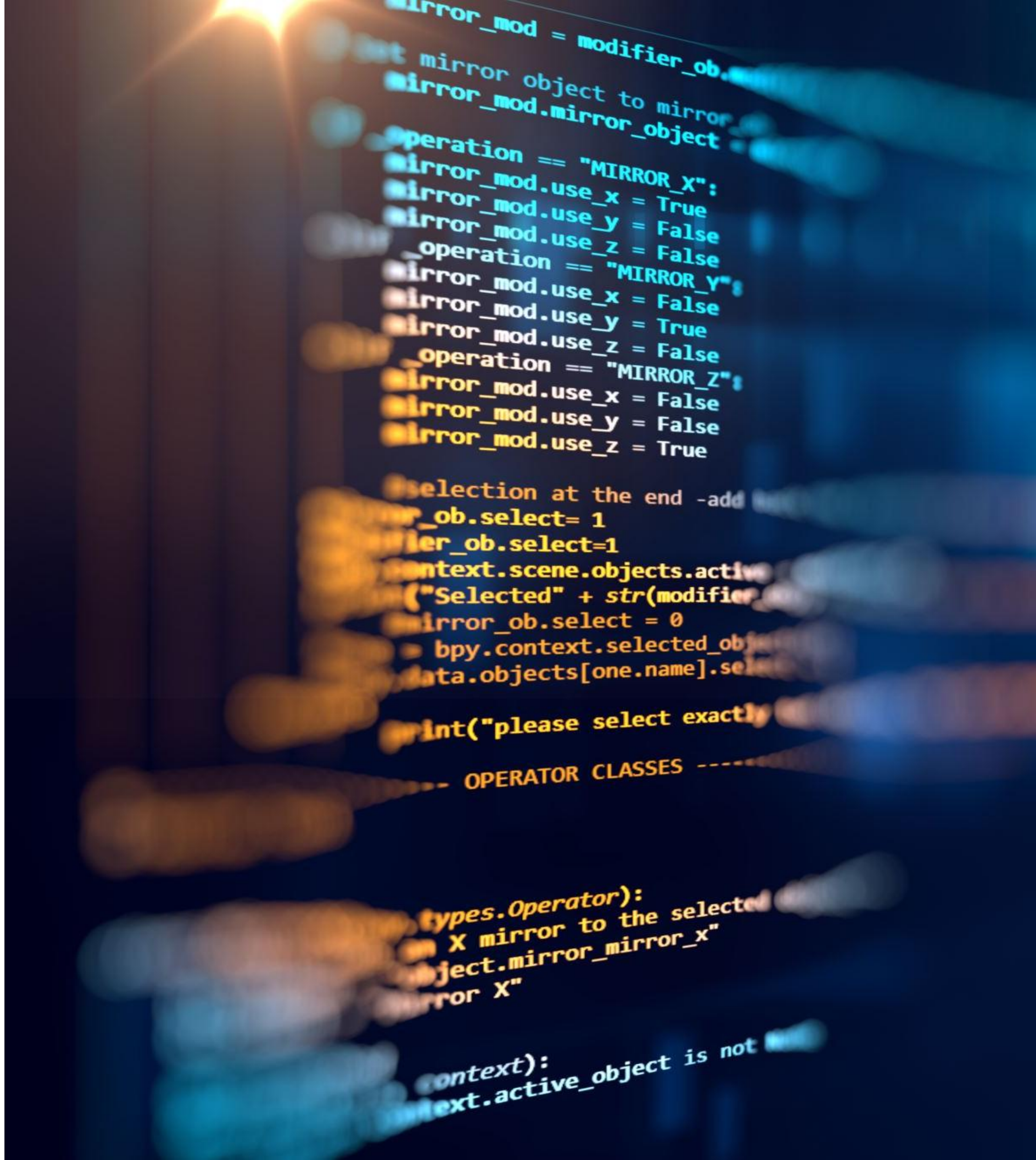
INFORMATION SYSTEMS 3A

Agile Software Development



Contents

- Agile methods
- Agile development techniques
- Agile project management
- Scaling agile methods



Plan-driven and agile development

Plan-driven development

- A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.

Agile development

- Specification, design, implementation and testing are interleaved and the outputs from the development process are decided through a process of negotiation during the software development process

Agile software development

- The agile software development methodology has recently become one of the most used software development techniques. Rather than the long-drawn-out release cycles in the previously popular waterfall methodology, the agile technique suggests regular short sprint release cycles.
- This allows the customers and stakeholders to have more involvement within the software development process. This helps promote a higher quality final product because it combats the difficult task of a customer fully understanding and identifying all requirements in the software project planning phase. This also allows for the stakeholders to adjust the priorities of remaining tasks easily throughout the entire software development process.

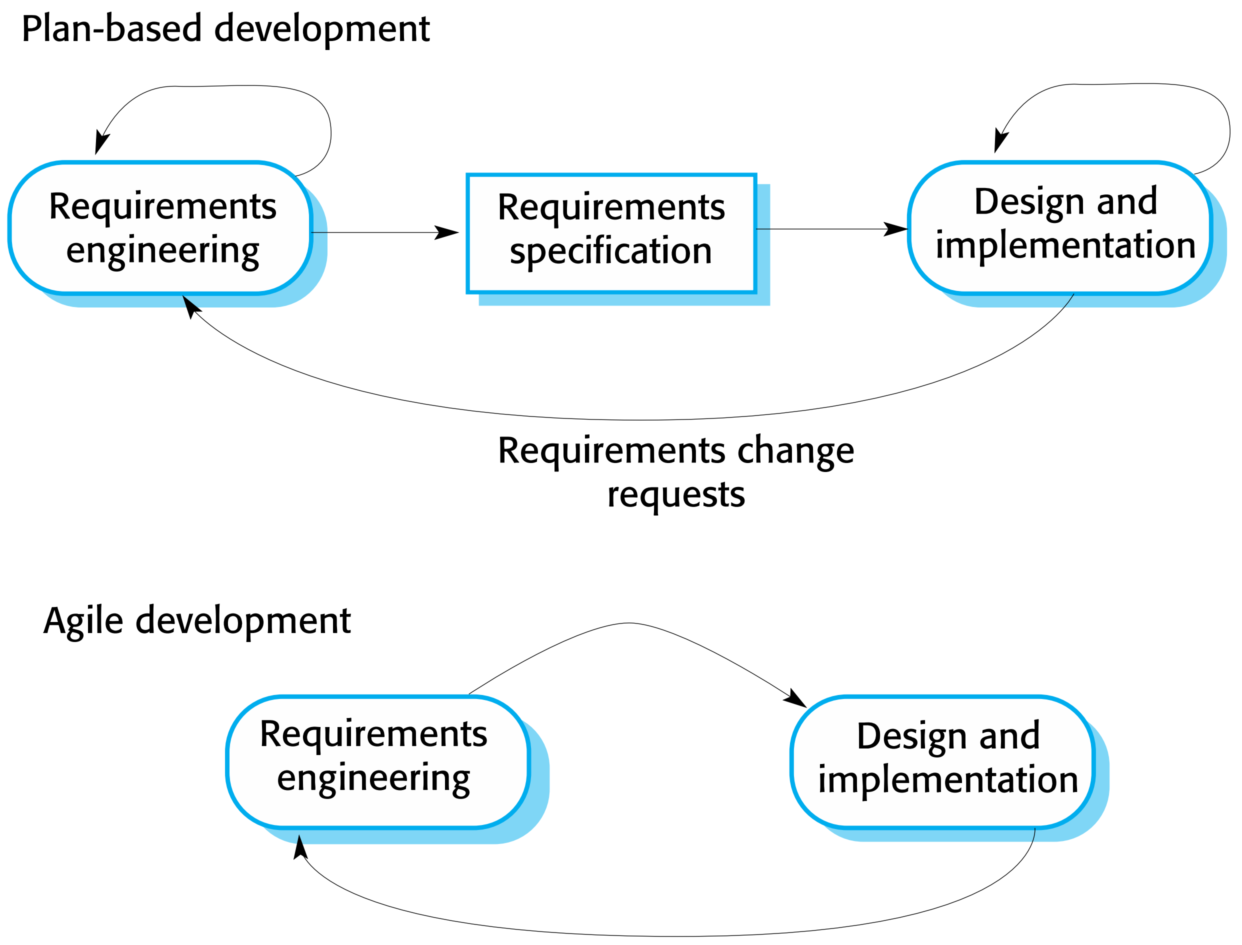
Agile software development

- 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 rapidly 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 software 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


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

Key characteristics of agile process models

- ➢Stronger focus on specification, design and implementation
- ➢Do enough documentation to move on
- ➢Incremental delivery
- ➢Client involvement
- ➢Dependency on automated tool support
- ➢Ability to respond to project change
- ➢Effective communication among all stakeholders
- ➢Drawing the customer into the team
- ➢Organizing a team so that it is in control of the work performed

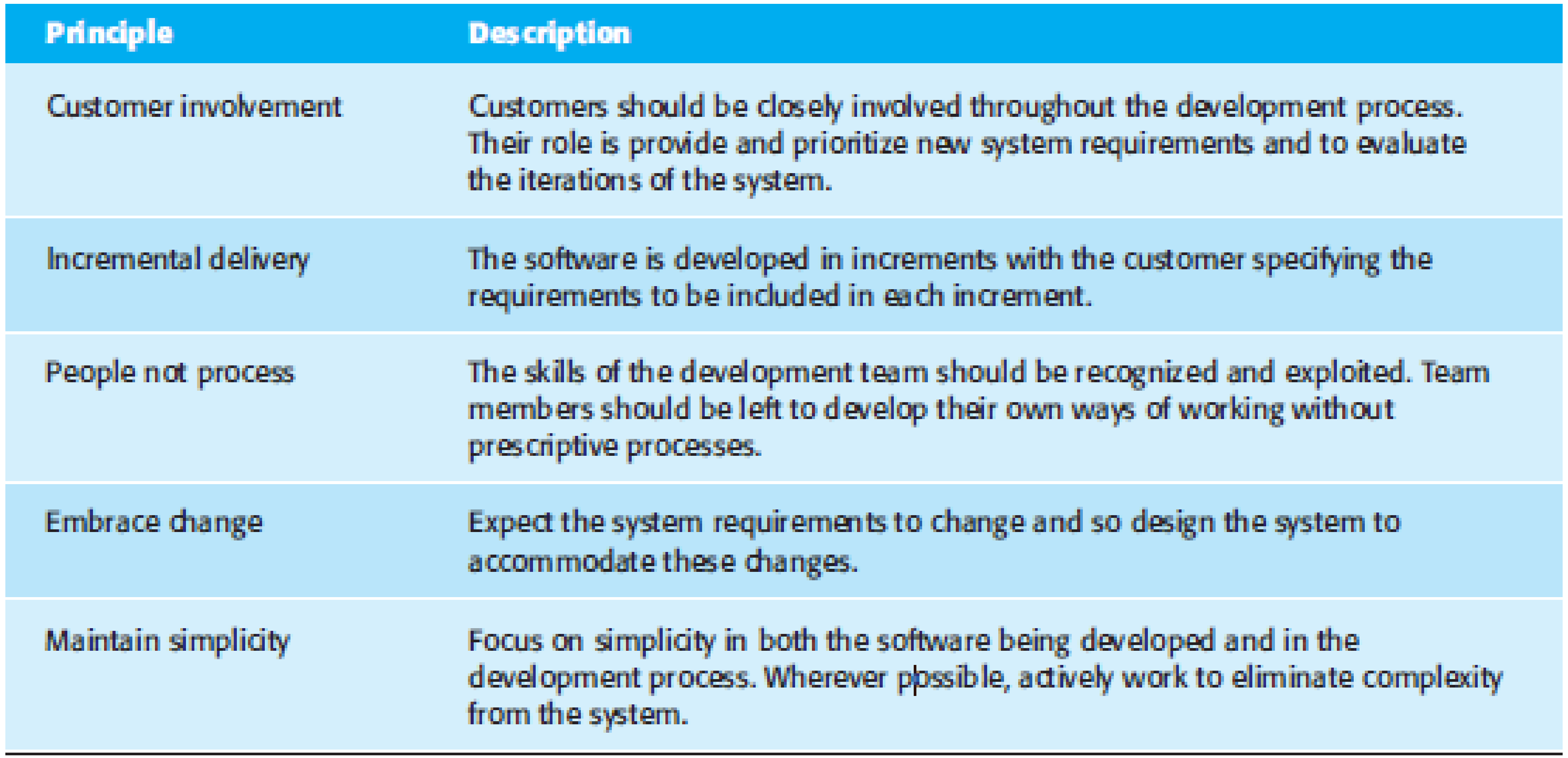
Agile manifesto

- 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 ppssible, actively work to eliminate complexity from the system.

Agile method applicability

- Product development where a software company is developing a small or medium-sized product for sale.
- 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.

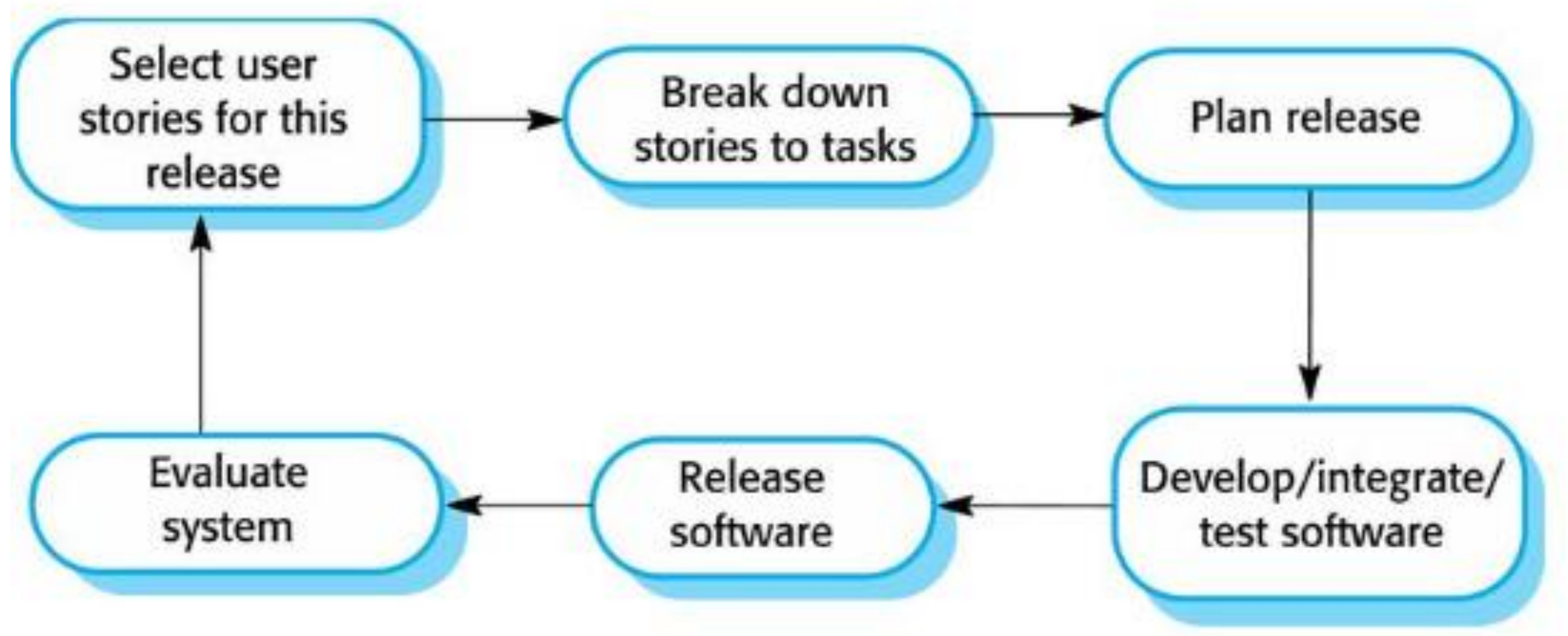
Agile development methods techniques

- Extreme Programming (XP)
- Scrum
- Kanban

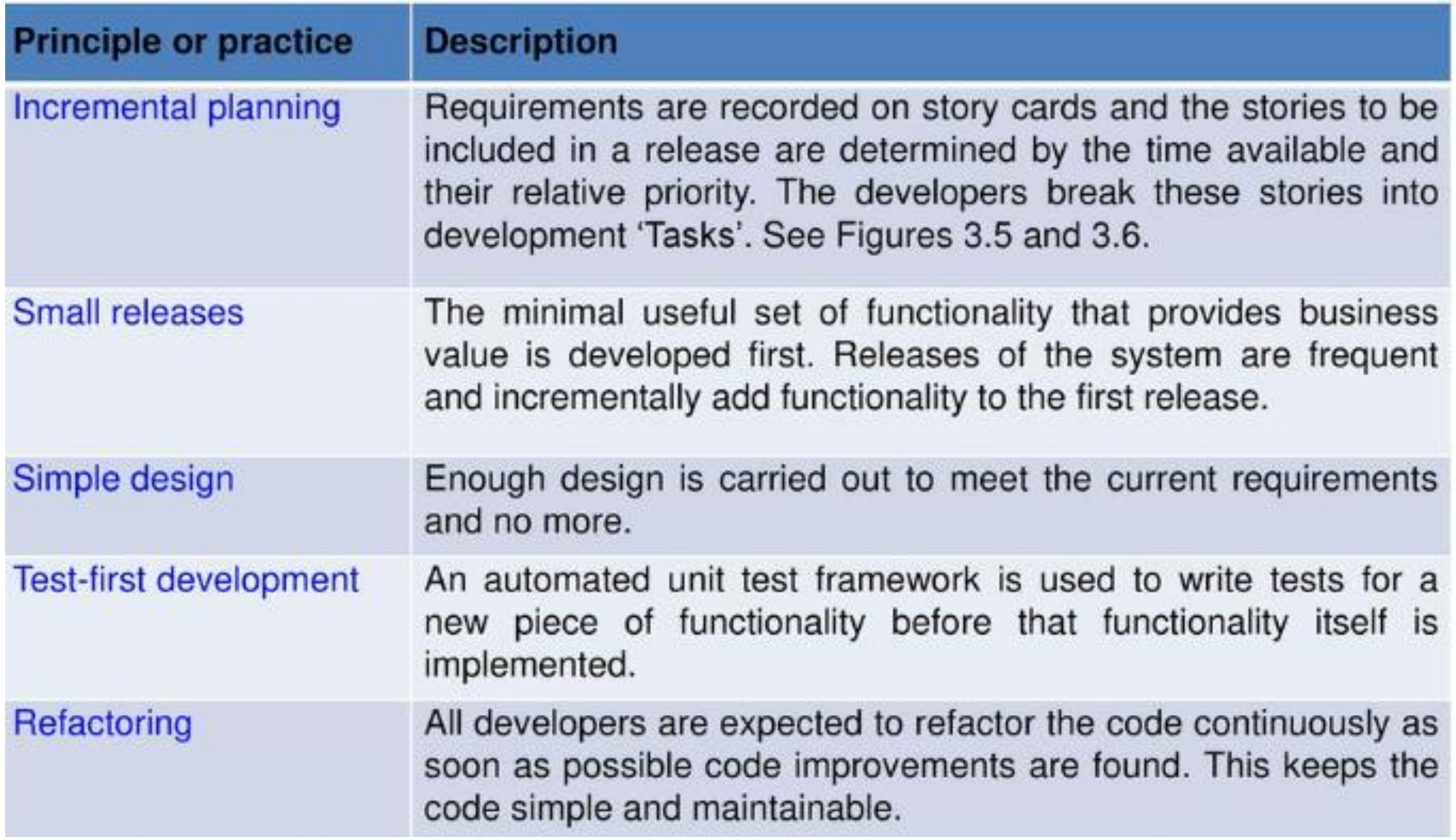
Extreme programming (XP)

- A very influential agile method, developed in the late 1990s, that introduced a range of agile development techniques.
- 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.

Extreme programming release cycle

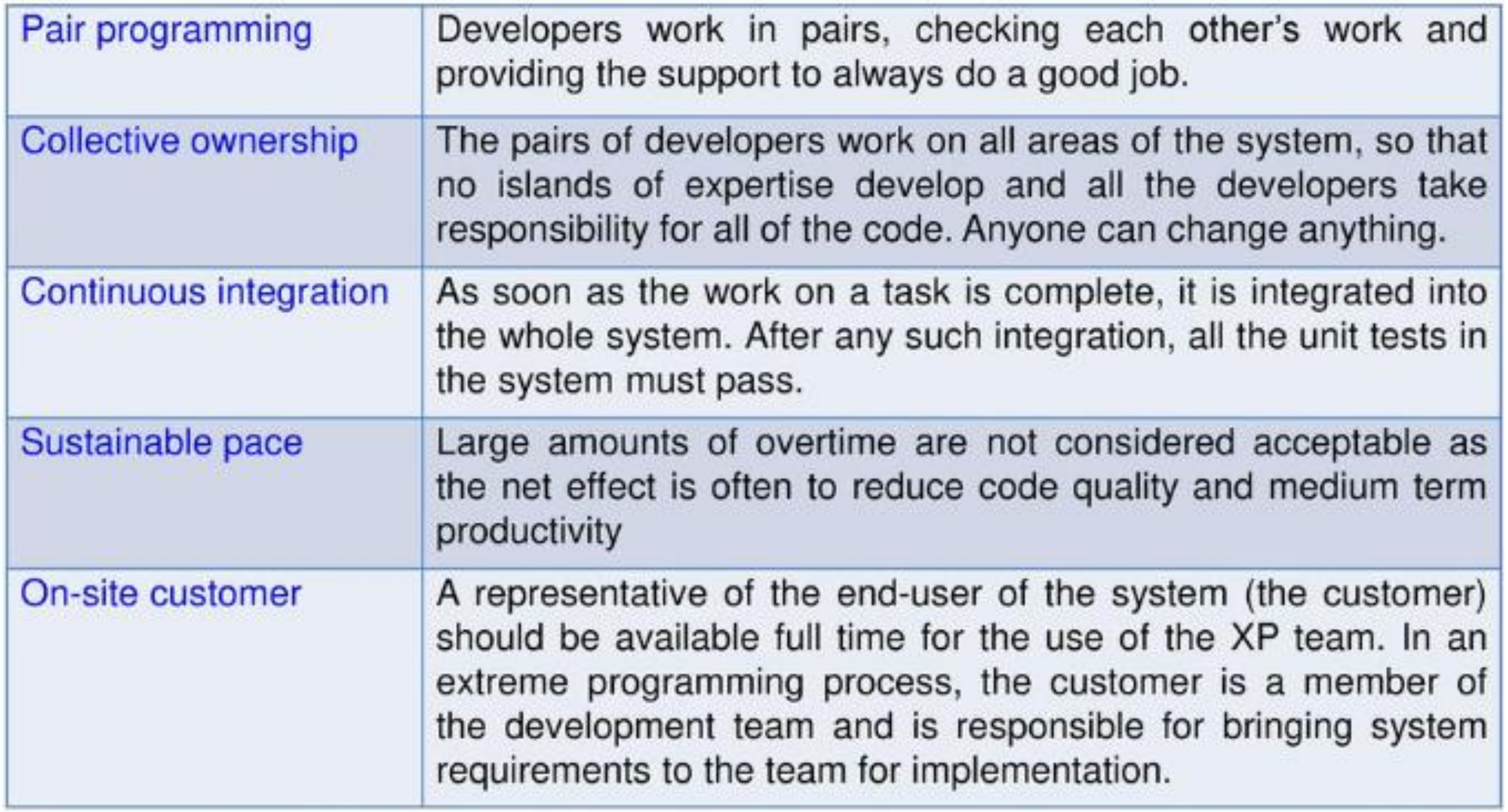

Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

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'. See Figures 3.5 and 3.6.
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.

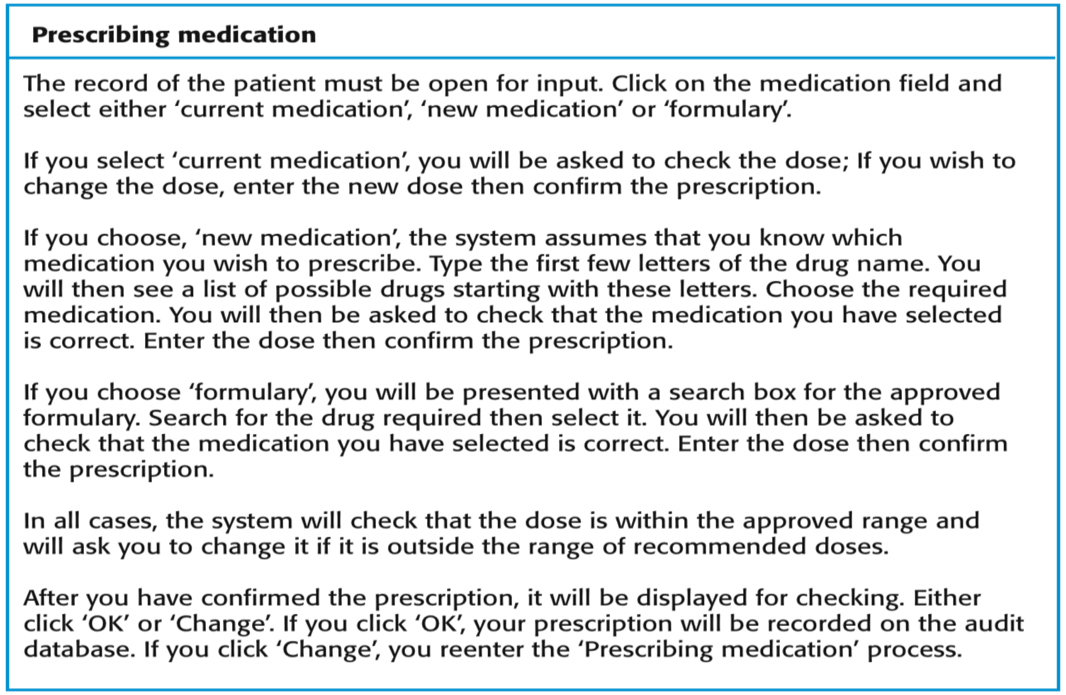
Influential XP practices

- 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 specification

- ✧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 scenarios or user stories.
- ✧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

The record of the patient must be open for input. Click on the medication field and select either ‘current medication’, ‘new medication’ or ‘formulary’.

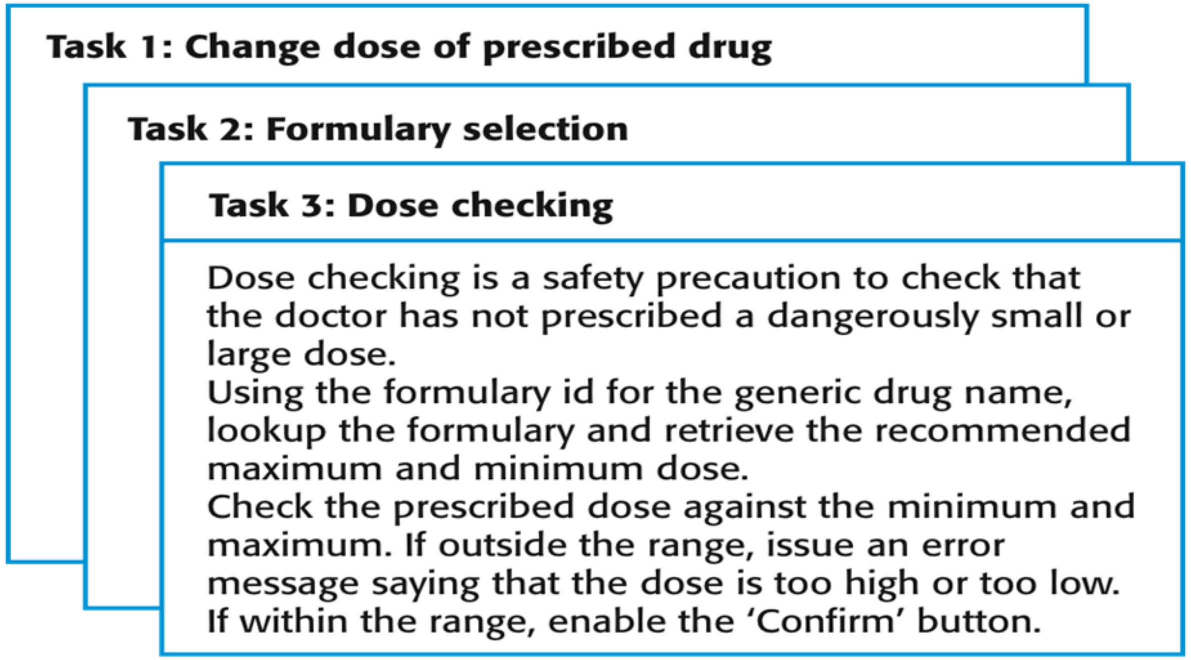
If you select ‘current medication’, you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, ‘new medication’, the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose ‘formulary’, you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click ‘OK’ or ‘Change’. If you click ‘OK’, your prescription will be recorded on the audit database. If you click ‘Change’, you reenter the ‘Prescribing medication’ process.



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 (Cleaning up the code)

- In Extreme Programming (XP), refactoring is the practice of continuously improving code without changing its external behavior. It helps keep the code clean, simple, and maintainable.

➢ More efficient

➢ Helps in finding the Bugs

➢ Remove duplicates

➢ Replacement of similar code sections

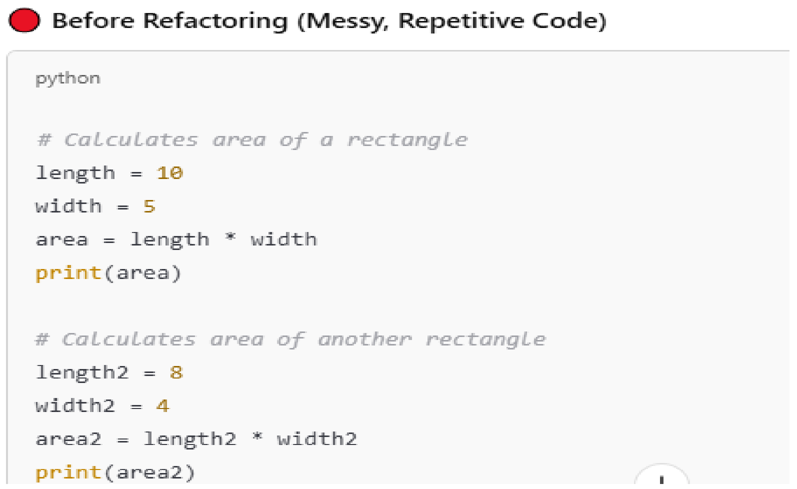
➢ Makes software easier to understand

➢ Improves the design of software

➢ Also helps in executing the program faster

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

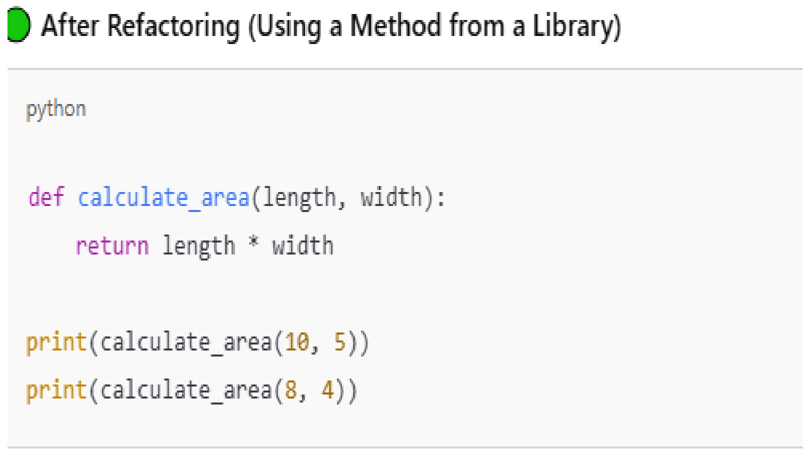


🔴 Before Refactoring (Messy, Repetitive Code)

python

```python
# Calculates area of a rectangle
length = 10
width = 5
area = length * width
print(area)

# Calculates area of another rectangle
length2 = 8
width2 = 4
area2 = length2 * width2
print(area2)
```



🟢 After Refactoring (Using a Method from a Library)

python

```python
def calculate_area(length, width):
    return length * width

print(calculate_area(10, 5))
print(calculate_area(8, 4))
```

Test-first 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
- ✧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.

Pair programming

- ✧In XP, programmers work in pairs, sitting together to develop code.
- ✧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 one person.
- ✧It encourages refactoring as the whole team can benefit from this.
- ✧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.

Scrum

- The Scrum approach is a general agile method, but its focus is 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

Roles in Scrum

There are three different roles needed in doing the scrum method, namely:

- **Scrum Master** : Scrum master is like a team coach who helps everyone follow the Scrum process smoothly. They **don’t manage the team** but make sure work happens without problems.
- **Product Owner**: Product Owner is the decision-maker in a Scrum team. They make sure the team is building the right product by setting priorities and defining features.The product owner can be a customer but might also be a product manager in a software company or other stakeholder representative.
- **Scrum Team**: Scrum team is a small, self-organizing group of people who work together to complete a project using the Scrum framework. It includes a mix of roles, with everyone working collaboratively to deliver value in short cycles (sprints).

The 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 who work with the customer to select the features and functionality to be developed during the sprint

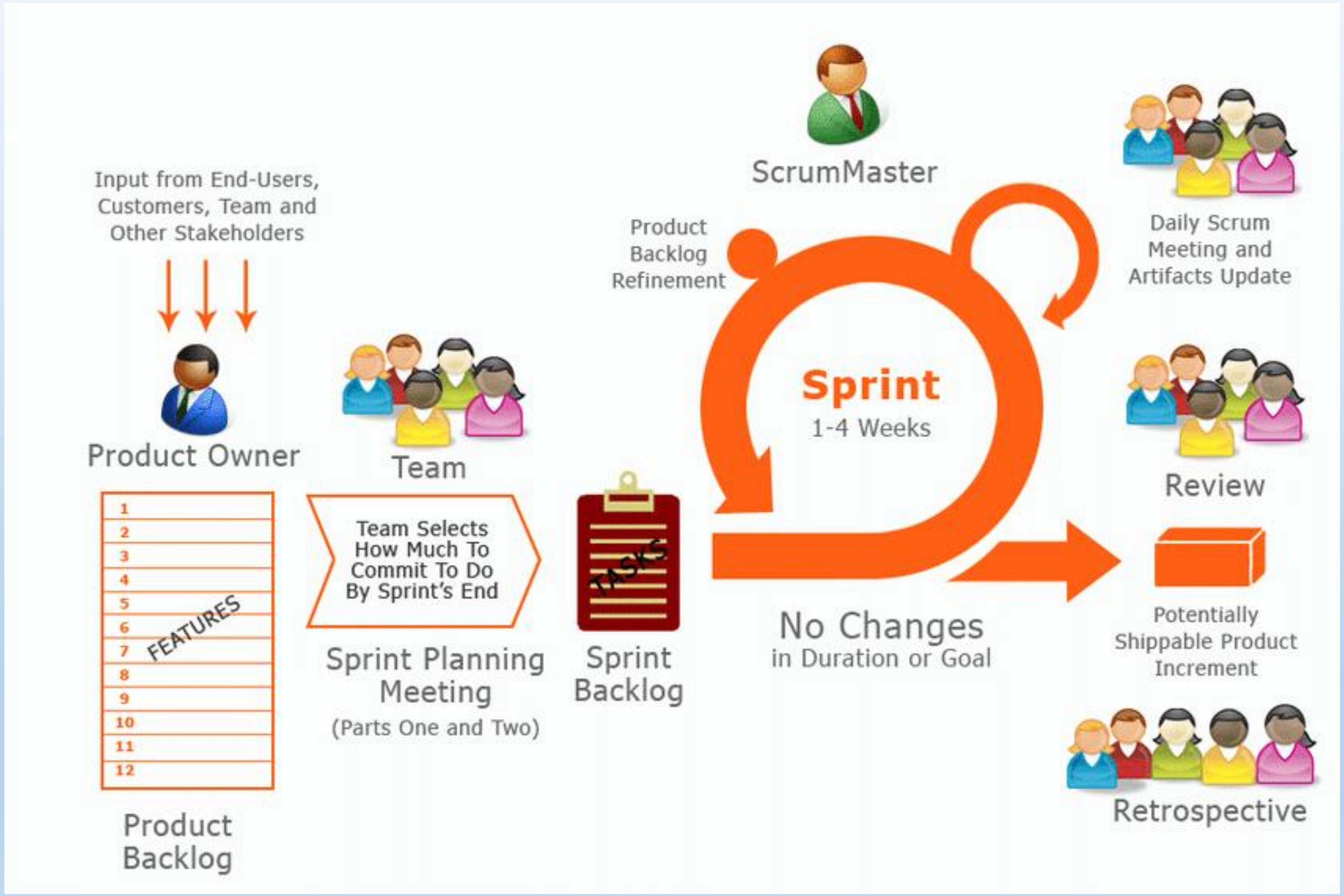
Teamwork in Scrum

- ✧The Scrum master is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team
- ✧The whole team attends short daily meetings which are referred as daily stand-up meeting where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day
  - This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them

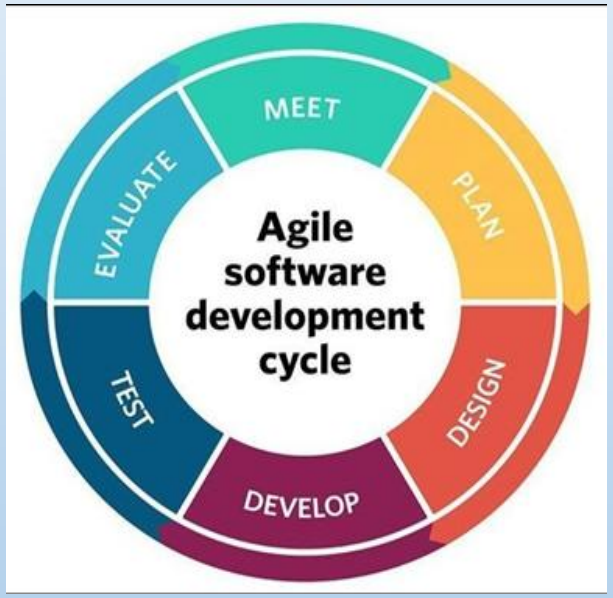
Scrum benefits

- ✧The product is broken down into a set of manageable and understandable chunks
- ✧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


Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Team
ScrumMaster
Product Backlog Refinement
Sprint
1-4 Weeks
Daily Scrum Meeting and Artifacts Update
Review
1
2
3
4
5
6
7
8
9
10
11
12
FEATURES
Team Selects How Much To Commit To Do By Sprint's End
TASKS
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Product Backlog
Sprint Planning Meeting
(Parts One and Two)
Sprint Backlog
Retrospective

PLANNING
PROGRAMMING
BETA TEST
HIGH LEVEL TEST
VALIDATION AND VERIFICATION
ANALYZE
DESIGN
Software Engineering


MEET
PLAN
DESIGN
DEVELOP
TEST
EVALUATE
Agile software development cycle

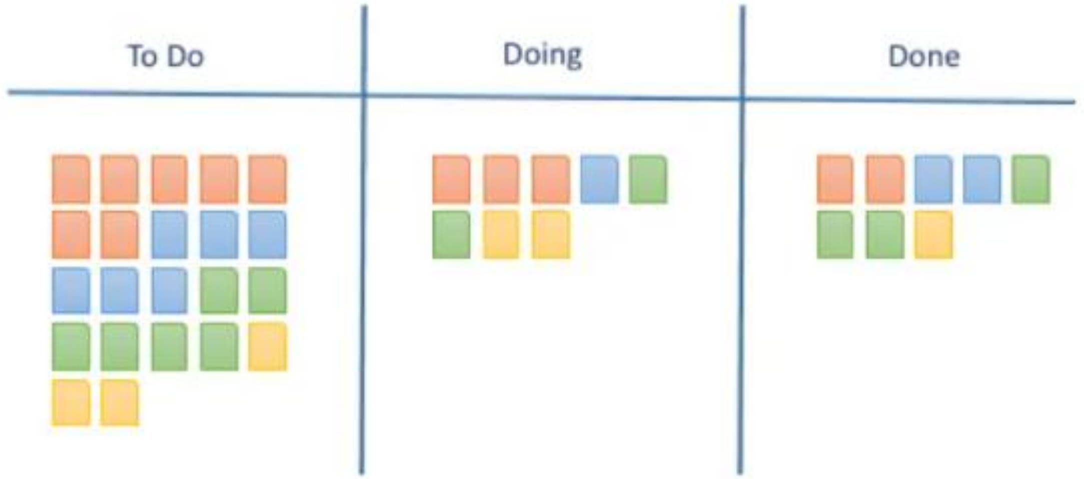
Kanban

- Kanban is an Agile framework that emphasizes visualization, workflow management, and continuous improvement.
- It involves creating a visual board that represents the work that needs to be done, the work that is in progress, and the work that has been completed.
- It is designed to help teams manage their work more efficiently by visualizing the workflow, limiting work in progress (WIP), and optimizing the flow of tasks or work items from start to finish.
- Kanban uses a kanban board to represent work items as cards that move through different stages of a workflow.

What is the difference between a Kanban board and a Kanban card?

- A **Kanban card** represents an individual task within a workflow.
  - Kanban cards include detailed information about each task, including its status, owner, and priority.
- A **Kanban board** is a visual tool that displays an entire workflow.
  - The Kanban board brings the Kanban cards together to give the team an overall view of a project.

Kanban Methodology



Four core principles make up the Kanban framework

- Start with what you know
- Pursue incremental change
- Respect the current process
- Encourage leadership at all levels

Kanban Key practices

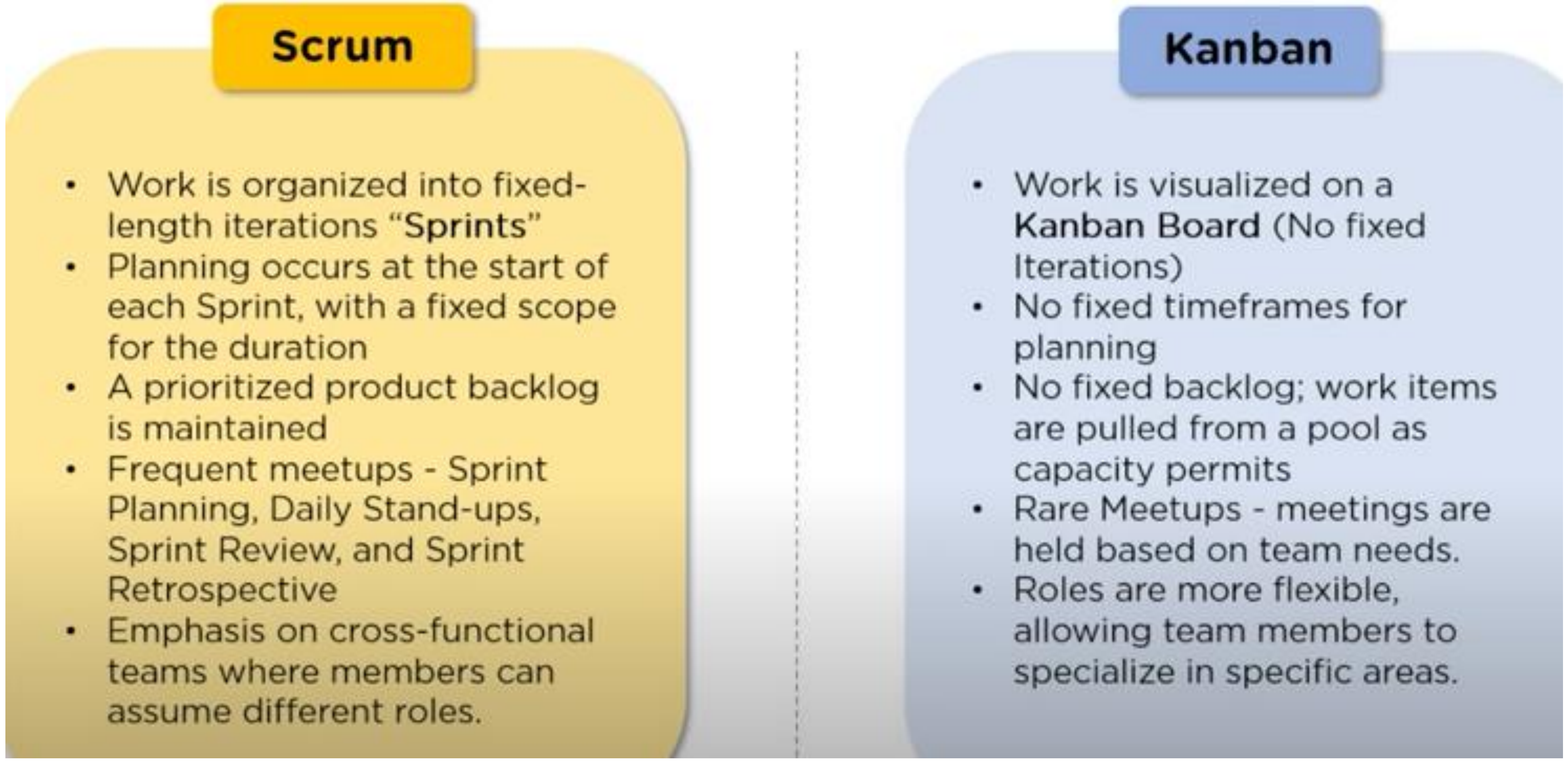
- **Visualize the workflow:** Make the process transparent and easy to understand.
- **Limit work-in-progress (WIP):** Reduce bottlenecks and improve flow by limiting the number of tasks in each stage.
- **Focus on flow:** Prioritize the continuous movement of work through the system.
- **Make explicit policies for the workflow:** Establish clear rules and procedures for how work moves through the process.
- **Implement feedback loops:** Continuously monitor and improve the process based on real-time data

Benefits of the Kanban method

The Kanban method offers a range of advantages that streamline project management and enhance team performance, including:

- **Improved workflow:** By visualizing tasks on a Kanban board, teams can identify bottlenecks and optimize workflows.
- **Increased efficiency:** Limiting WIP ensures that team members focus on completing tasks instead of feeling overwhelmed with multitasking.
- **Enhanced collaboration:** The visual nature of Kanban boards fosters a collaborative environment, as everyone is aware of their responsibilities and the status of each task.

Scrum Vs Kanban



Scrum

- Work is organized into fixed-length iterations "**Sprints**"
- Planning occurs at the start of each Sprint, with a fixed scope for the duration
- A prioritized product backlog is maintained
- Frequent meetups - Sprint Planning, Daily Stand-ups, Sprint Review, and Sprint Retrospective
- Emphasis on cross-functional teams where members can assume different roles.

Kanban

- Work is visualized on a **Kanban Board** (No fixed Iterations)
- No fixed timeframes for planning
- No fixed backlog; work items are pulled from a pool as capacity permits
- Rare Meetups - meetings are held based on team needs.
- Roles are more flexible, allowing team members to specialize in specific areas.

Scaling agile methods

- ✧Agile methods have proved to be successful for small and medium sized projects that can be developed by a small co-located team
- ✧It is sometimes argued that the success of these methods comes because of improved communications which is possible when everyone is working together
- ✧Scaling up agile methods involves changing these to cope with larger, longer projects where there are multiple development teams, perhaps working in different locations

Scaling out and scaling up

- **'Scaling up'** is concerned with using agile methods for developing large software systems that cannot be developed by a small team
- **'Scaling out'** is concerned with how agile methods can be introduced across a large organization with many years of software development experience
- When scaling agile methods, it is important to maintain agile fundamentals:
  - Flexible planning, frequent system releases, continuous integration, test-driven development, and good team communication

Agile across organizations

Some barriers to agile adoption:

- ➢Project managers are largely inexperienced with agile
- ➢Bureaucratic procedures that are necessary for quality
- ➢Needs highly and multi-skilled team members
- ➢Well ingrained plan-driven processes

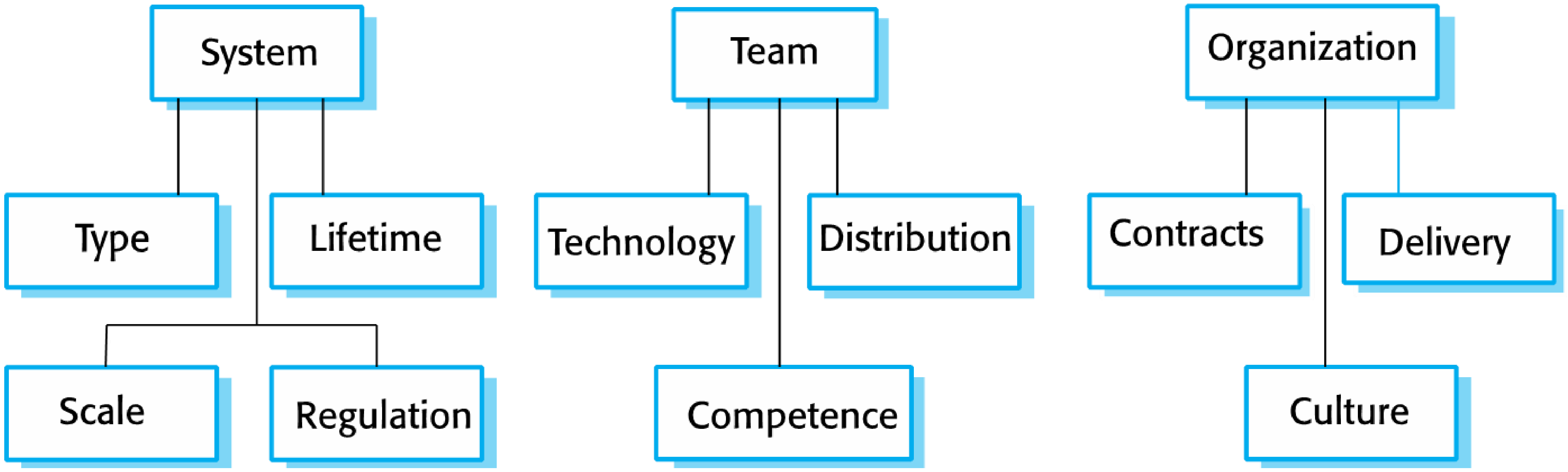
Agile maintenance

Key problems are:

- Lack of product documentation
- Keeping customers involved in the development process
- Maintaining the continuity of the development team

Agile or Plan Driven(How to decide?)

- What factors should the team consider before choosing?



Agile methods across organizations

- Project managers who do not have experience of agile methods may be reluctant to accept the risk of a new approach.
- Large organizations often have quality procedures and standards that all projects are expected to follow, and, because of their bureaucratic nature, these are likely to be incompatible with agile methods.
- Agile methods seem to work best when team members have a relatively high skill level. However, within large organizations, there are likely to be a wide range of skills and abilities.
- There may be cultural resistance to agile methods, especially in those organizations that have a long history of using conventional systems engineering processes.