Unit 3 – Agile Software Development（下）
Topics covered

✧ Agile methods

✧ Agile development techniques

✧ Agile project management

✧ Scaling agile methods

Agile project management

- The principal responsibility of software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project.
- The standard approach to project management is plan-driven. Managers draw up a plan for the project showing what should be delivered, when it should be delivered and who will work on the development of the project deliverables.
- Agile project management requires a different approach, which is adapted to incremental development and the practices used in agile methods.

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.
-

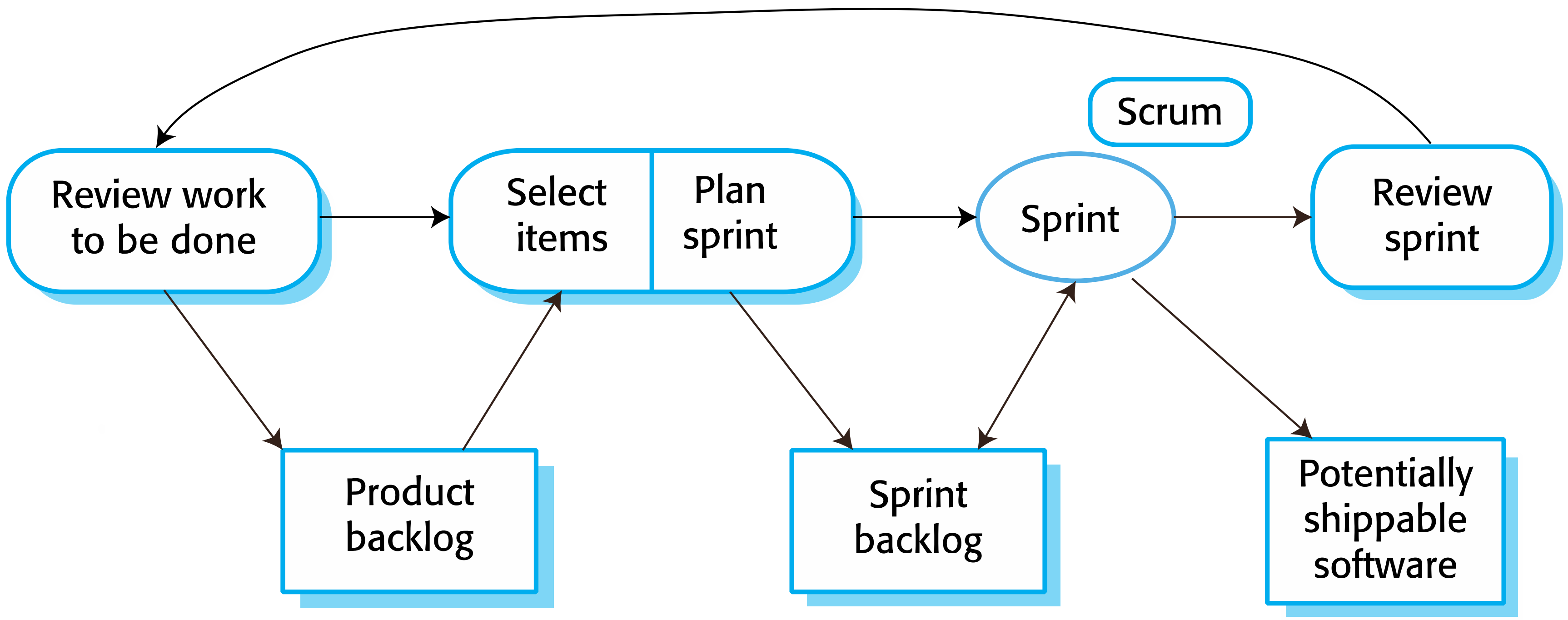
Scrum terminology (a)

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



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 from the product backlog to be developed during the sprint.

The Sprint cycle

- ✧ Once these are agreed, the team organize themselves to develop the software.
- ✧ During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called ‘Scrum master’.
- ✧ The role of the Scrum master is to protect the development team from external distractions.
- ✧ At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

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 (Scrums) 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.
- ✧ 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.

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 importaant to maintain agile fundamentals:
  - Flexible planning, frequent system releases, continuous integration, test-driven development and good team communications.

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 my many legal departments because what has to be delivered cannot be guaranteed.

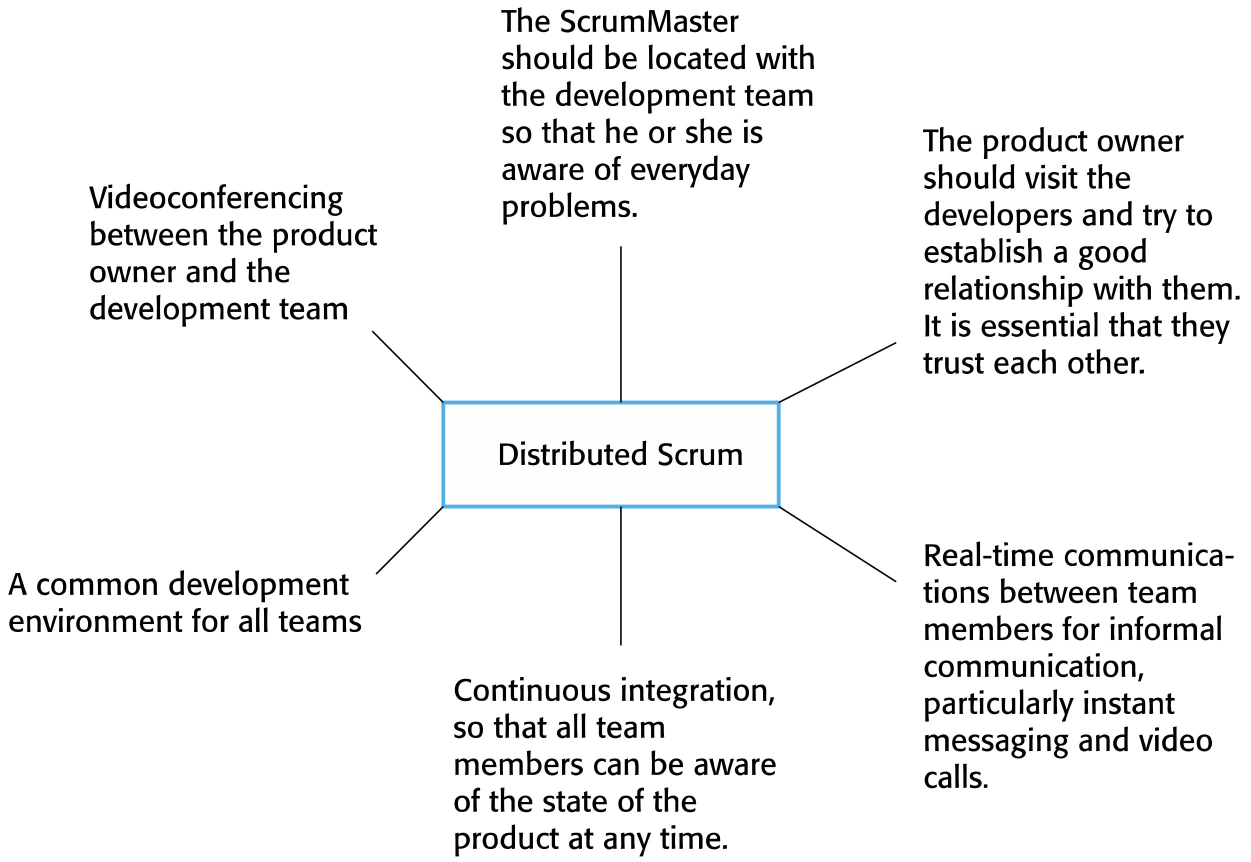
Agile methods and software maintenance

- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues:
  - Are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
  - Can agile methods be used effectively for evolving a system in response to customer change requests?
- Problems may arise if original development team cannot be maintained.

Agile maintenance

- ✧ Key problems are:
  - Lack of product documentation
  - Keeping customers involved in the development process
  - Maintaining the continuity of the development team
- ✧ Agile development relies on the development team knowing and understanding what has to be done.
- ✧ For long-lifetime systems, this is a real problem as the original developers will not always work on the system.

Distributed Scrum


The ScrumMaster should be located with the development team so that he or she is aware of everyday problems.
Videoconferencing between the product owner and the development team
The product owner should visit the developers and try to establish a good relationship with them. It is essential that they trust each other.
Distributed Scrum
A common development environment for all teams
Real-time communica-tions between team members for informal communication, particularly instant messaging and video calls.
Continuous integration, so that all team members can be aware of the state of the product at any time.

Agile and plan-driven methods

- Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:
  - 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? If so, consider using agile methods.
  - How large is the system that is being developed? Agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.

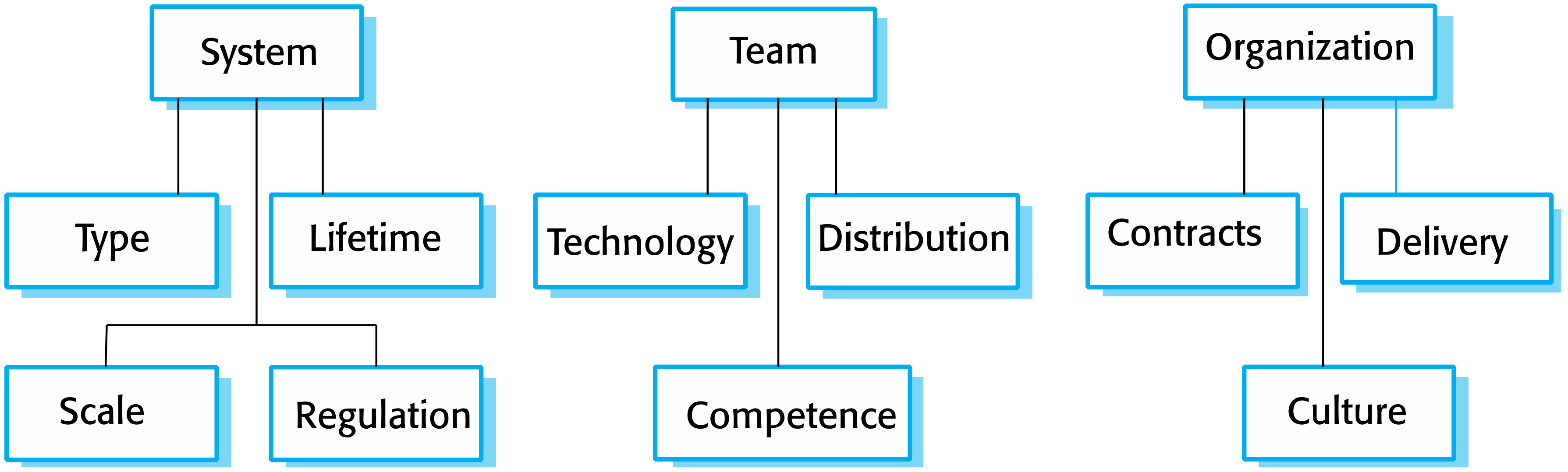
Agile principles and organizational practice

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.

Agile principles and organizational practice

Principle	Practice
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.

People and teams

- **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 dsitributed.
- What support **technologies** are available?
  - IDE support for visualisation and program analysis is essential if design documentation is not available.

Organizational issues

- ✧ **Traditional** engineering organizations have a **culture** of **plan-based** development, as this is the norm in engineering.
- ✧ Is it standard organizational practice to develop a **detailed system specification**?
- ✧ Will customer **representatives** be available to provide **feedback** of system increments?
- ✧ Can **informal** agile development **fit into** the **organizational culture** of detailed documentation?

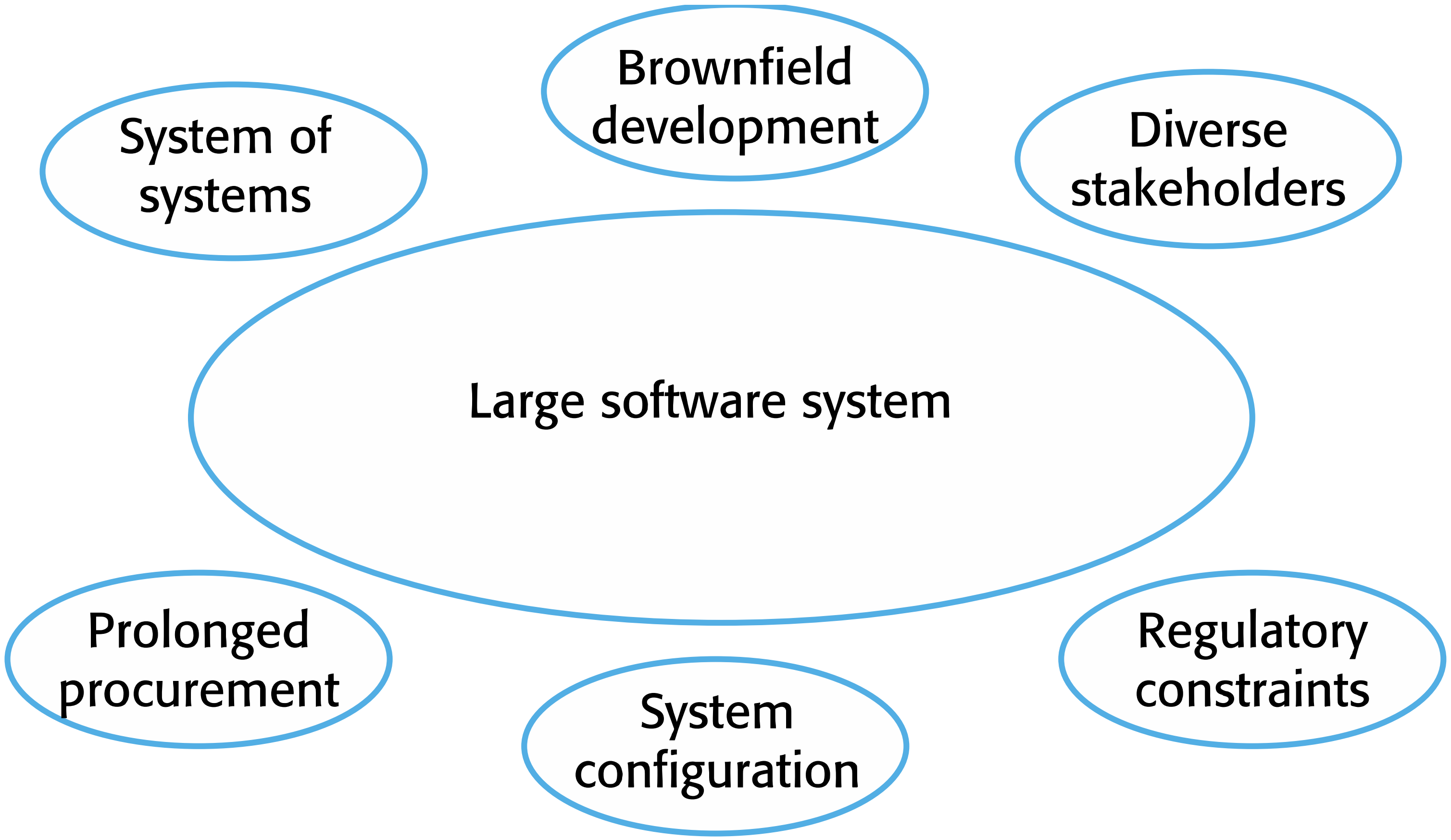
Agile methods for large systems

- ✧ Large systems are usually collections of separate, communicating systems, where separate teams develop each system. Frequently, these teams are working in different places, sometimes in different time zones.
- ✧ Large systems are 'brownfield systems', that is they include and interact with a number of existing systems. Many of the system requirements are concerned with this interaction and so don't really lend themselves to flexibility and incremental development.
- ✧ Where several systems are integrated to create a system, a significant fraction of the development is concerned with system configuration rather than original code development.

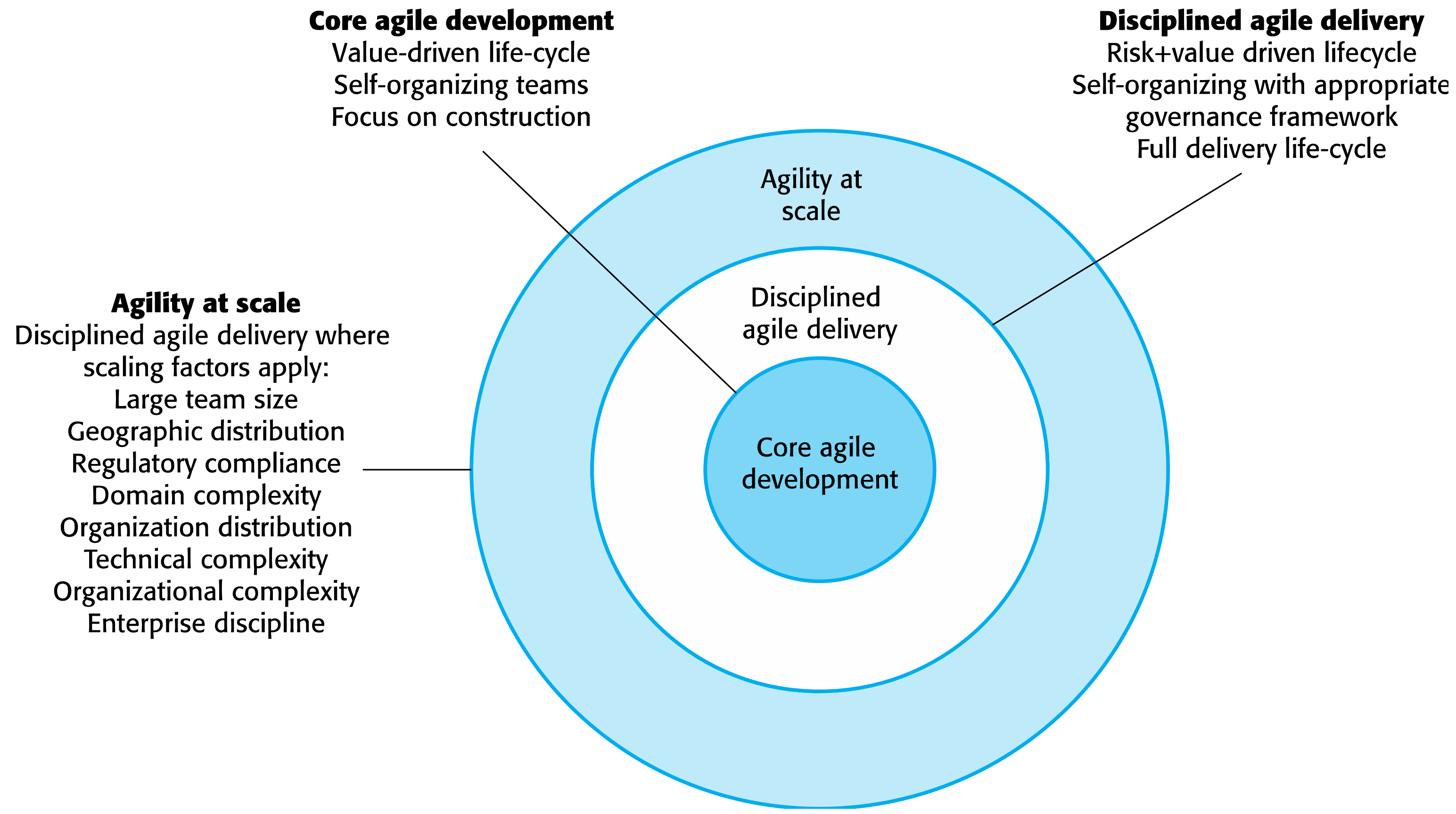
Large system development

- ✧ Large systems and their development processes are often constrained by external rules and regulations limiting the way that they can be developed.
- ✧ Large systems have a long procurement and development time. It is difficult to maintain coherent teams who know about the system over that period as, inevitably, people move on to other jobs and projects.
- ✧ Large systems usually have a diverse set of stakeholders. It is practically impossible to involve all of these different stakeholders in the development process.

Factors in large systems



IBM's agility at scale model


Core agile development
Value-driven life-cycle
Self-organizing teams
Focus on construction
Disciplined agile delivery
Risk+value driven lifecycle
Self-organizing with appropriate governance framework
Full delivery life-cycle
Agility at scale
Agility at scale
Disciplined agile delivery where scaling factors apply:
Large team size
Geographic distribution
Regulatory compliance
Domain complexity
Organization distribution
Technical complexity
Organizational complexity
Enterprise discipline
Disciplined agile delivery
Core agile development

Scaling up to large systems

- A completely incremental approach to requirements engineering is impossible.
- There cannot be a single product owner or customer representative.
- For large systems development, it is not possible to focus only on the code of the system.
- Cross-team communication mechanisms have to be designed and used.
- Continuous integration is practically impossible. However, it is essential to maintain frequent system builds and regular releases of the system.

Multi-team Scrum

- *Role replication*
  - Each team has a Product Owner for their work component and ScrumMaster.
- *Product architects*
  - Each team chooses a product architect and these architects collaborate to design and evolve the overall system architecture.
- *Release alignment*
  - The dates of product releases from each team are aligned so that a demonstrable and complete system is produced.
- *Scrum of Scrums*
  - There is a daily Scrum of Scrums where representatives from each team meet to discuss progressand plan work to be done.

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.

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 include
  - User stories for system specification
  - Frequent releases of the software,
  - Continuous software improvement
  - Test-first development
  - Customer participation in the development team.

Key points

- ✧ 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.
- ✧ Scaling agile methods for large systems is difficult.
  - ▪ Large systems need up-front design and some documentation and organizational practice may conflict with the informality of agile approaches.