Chapter four

Software engineering
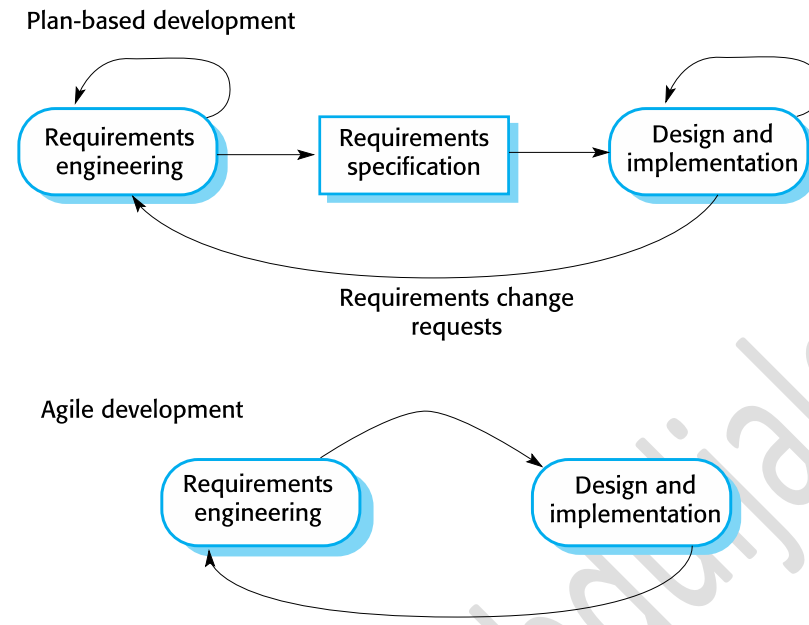
Agile Software Development

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

Plan-based development

Requirements engineering → Requirements specification → Design and implementation

Requirements change requests

Agile development

Requirements engineering → Design and implementation

◇ 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.

- Not necessarily waterfall model – plan-driven, incremental development is possible

- Iteration occurs within activities.

◇ Agile development

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

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:
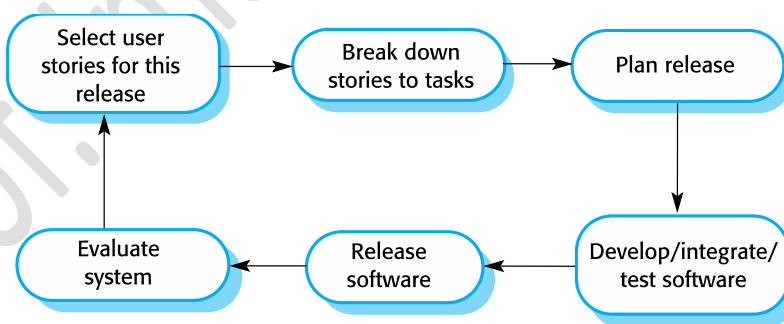
- Focus on the code rather than the design

- 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 development techniques

Extreme programming

✧ 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.

The 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 → (back to Select user stories)

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.

- ✧ Key practices

  - ▪ User stories for specification

  - ▪ Refactoring

  - ▪ Test-first development

  - ▪ Pair programming

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.

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.

Prof. Iman Qays Abduljaleel

- ✧ 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 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.

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.

Prof. Iman Qays Abduljaleel

✧ 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