AGILE METHODS - BACKGROUND

SOFTWARE ENGINEERING 1

SOON PHEI, TIN

CONTENT

- Agile methods
- Plan-driven and iterative (agile development)
- Scrum Framework

BACKGROUND

- Businesses now operate in a global, rapidly changing environment. They must respond to new opportunities and markets, changing economic conditions, and the emergence of competing products and services

What is the implication for software development?

BACKGROUND

- In fact, many businesses are willing to trade off software quality and compromise on requirements to achieve faster deployment of the software that they need.
- Impossible to derive a complete set of stable software requirements. The initial requirements inevitably change.
- Deliver the system ASAP to allow the users to gain experience and discover clearer real requirements.
- The requirements are likely to change quickly and unpredictably due to external factors.

BACKGROUND

- Software processes that plan on completely specifying the requirements and then designing, building, and testing the system are not geared to rapid software development.
- As the requirements change or as requirements problems are discovered, the system design or implementation must be reworked and retested.

BACKGROUND

- IBM introduced incremental development in the 1980s (Mills et al., 1980).
- The introduction of so-called fourth generation languages, also in the 1980s, supported the idea of quickly developing and delivering software (Martin, 1981).
- Late 1990s with the development of the notion of agile approaches such as
  - DSDM (Stapleton, 1997)
  - Scrum (Schwaber and Beedle, 2001)
  - extreme programming (Beck, 1999; Beck, 2000).

AGILE METHODS – RAPID SOFTWARE DEVELOPMENT

SOFTWARE ENGINEERING 1

SOON PHEI, TIN

RAPID SOFTWARE DEVELOPMENT

- The software is not developed as a single unit but as a series of increments, with each increment including new system functionality.
- Fundamental characteristics:
  - The processes are interleaved.
  - Minimum documentation
  - Developed in a series of versions, or increments, with system stakeholders involvement.
  - System user interfaces are often developed using an interactive development system that allows the interface design to be quickly created

RAPID SOFTWARE DEVELOPMENT

- Other approaches to RAD includes: -
  - Adaptive software development
  - Agile methodologies
  - Spiral model
  - Unified software development process

PLAN-DRIVEN DEVELOPMENT APPROACH

- Large, long lived software - careful project planning, formalized quality assurance, the use of analysis and design methods supported by CASE tools, and controlled and rigorous software development processes

What are some examples of this type of software?

AGILE METHODS

- Agile allowed the development team to focus on the software itself rather than on its design and documentation
- universally rely on an incremental approach to software specification, development, and delivery.
- They are best suited to application development where the system requirements usually change rapidly during the development process
- They are intended to deliver working software quickly to customers, who can then propose new and changed requirements to be included in later iterations of the system.

AGILE METHODS

- The philosophy behind agile methods is reflected in the agile manifesto that was agreed on by many of the leading developers of these methods.
  - *Individuals and interactions over processes and tools*
  - *Working software over comprehensive documentation*
  - *Customer collaboration over contract negotiation*
  - *Responding to change over following a plan*

AGILE METHODS

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

AGILE METHODS

- Some of the agile methods
  - Extreme programming (Beck, 1999; Beck, 2000)
  - Scrum (Cohn, 2009; Schwaber, 2004; Schwaber and Beedle, 2001)
  - Crystal (Cockburn, 2001; Cockburn, 2004)
  - Adaptive Software Development (Highsmith, 2000)
  - DSDM (Stapleton, 1997; Stapleton, 2003)
  - Feature Driven Development (Palmer and Felsing, 2002)
- These agile methods are all based around the notion of incremental development and delivery with different processes
- However, they share a set of principles, based on the agile manifesto, and so have much in common.

AGILE METHODS – THE CHALLENGES

- In practice, the principles underlying agile methods are sometimes difficult to realize:
  - Its success depends on having a customer who is willing and able to spend time with the development team and who can represent all system stakeholders
  - Individual team members may not have suitable personalities for the intense involvement
  - Prioritizing changes can be extremely difficult, especially in systems for which there are many stakeholders

AGILE METHODS – MORE CHALLENGES

- Maintaining simplicity requires extra work
- It is difficult for some organization to accept informal processes defined by development teams
- The software requirements document is usually part of the contract between the customer and the supplier (software company)

AGILE METHODS – MORE CHALLENGES

- Because incremental specification is inherent in agile methods, writing contracts for this type of development may be difficult.
- Agile methods must rely on contracts in which the customer pays for the time required for system development rather than the development of a specific set of requirements
- If problems arise then there may be difficult disputes over who is to blame and who should pay for the extra time and resources required to resolve the problems.

AGILE METHODS – MAINTENANCE

- There are two questions that should be considered when considering agile methods and maintenance:
  - 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?
- Formal documentation is supposed to ease system evolution and maintenance. In practice, however, formal documentation is often not kept up to date and so does not in system maintainability

AGILE METHODS – MAINTENANCE

- Agile methods supporters argue that it is a waste of time to write this out-of-date documentation
- The key to implementing maintainable software is to produce high-quality, readable code.
- The key document is the system requirements document, which tells the software engineer what the system is supposed to do

AGILE METHODS – MAINTENANCE

- The main difficulty after software delivery is likely to be keeping customers involved in the process.
- The other problem that is likely to arise is maintaining continuity of the development team. Agile methods rely on team members understanding aspects of the system without having to consult documentation.
- If an agile development team is broken up, then this implicit knowledge is lost and it is difficult for new team members to build up the same understanding of the system and its components.

PLAN-DRIVEN AND AGILE DEVELOPMENT

- Most software projects include practices from plan-driven and agile approaches.
- To decide on the balance between a plan-based and an agile approach, you must answer a range of technical, human, and organizational questions:

  - Detail specification and design needed?
  - Is incremental strategy realistic?
  - How large is the system?
  - What type of system being developed?
  - System life span?
  - Available technologies and tools?
  - Organization of the team?
  - Cultural issues?
  - Available skillsets?
  - External regulation?

THE END
THANK YOU