These process models are centered around responsiveness to change.

AGILE PROCESS MODELS

Agile Methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods.
- 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

- *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.*

Agile Software Development Principles

- ✓ Customer satisfaction by rapid, continuous delivery of useful software
- ✓ Even late changes in requirements are welcomed
- ✓ Working software is delivered frequently (weeks rather than months)
- ✓ Close, daily cooperation between business people and developers
- ✓ Projects are built around motivated individuals, who should be trusted
- ✓ Face-to-face conversation is the best form of communication (co-location)

Agile Software Development Principles

- ✓ Working software is the principal measure of progress
- ✓ Agile processes promote sustainable development
- ✓ Continuous attention to technical excellence and good design
- ✓ Simplicity is essential
- ✓ Best architectures, requirements, and designs emerge from self-organizing teams
- ✓ Regular adaptation to changing circumstances

Problems with Agile Methods

- ! It can be difficult to keep the interest of customers who are involved in the process.
- ! Team members may be unsuited to the intense involvement that characterizes agile methods.
- ! Prioritizing changes can be difficult where there are multiple stakeholders.
- ! Maintaining simplicity requires extra work.
- ! Contracts may be a problem as with other approaches to iterative development.

Agile Methods and Software Maintenance

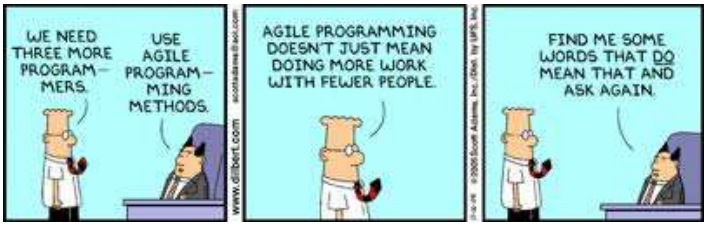
- If agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues:
  - Are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
  - Can agile methods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

Technical, Human, Organizational Issues

- Is it important to have a very detailed specification and design before moving to implementation?
- Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic?
- How large is the system that is being developed?

Technical, Human, Organizational Issues

- What type of system is being developed?
- What is the expected system lifetime?
- What technologies are available to support system development?
- How is the development team organized?
- Are there cultural or organizational issues that may affect the system development?
- How good are the designers and programmers in the development team?
- Is the system subject to external regulation?



AGILE MODELS

Extreme Programming

- Perhaps the best-known and most widely used agile method.
- Extreme Programming (XP) takes an 'extreme' approach to iterative development.

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.

The Extreme Programming Release Cycle

XP Practices

- Incremental planning
- Small releases
- Simple design
- Test-first development
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration
- Sustainable pace
- On-site customer

Requirements Scenarios

- Customer makes decisions on requirements.
- User requirements are expressed as scenarios or user stories.
- Are written on cards and the development team break them down into implementation tasks.
- 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.

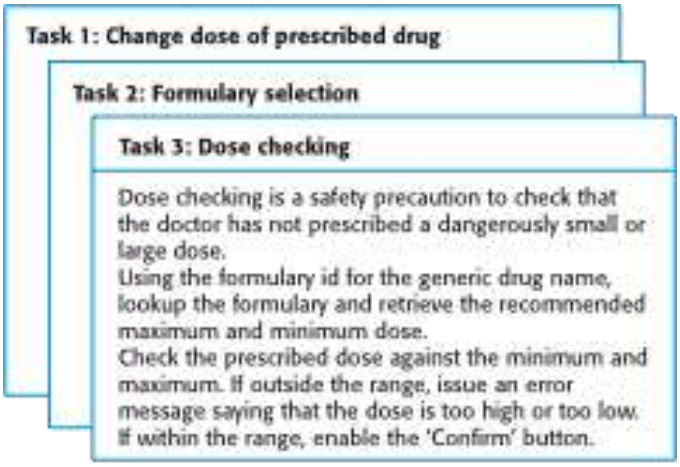
If you choose, ‘new medication’, the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication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

Examples of Task Cards for Prescribing Medication



XP and Change

- 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.
- However, some changes requires architecture refactoring and this is much more expensive.

Testing in XP

- 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.

Test Case Description for Dose Checking

Test 4: Dose checking

Input:
1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:
1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:
OK or error message indicating that the dose is outside the safe range.

Test Automation

- Test automation means that tests are written as executable components before the task is implemented
- As testing is automated, there is always a set of tests that can be quickly and easily executed

XP Testing Difficulties

- Programmers prefer programming to testing and sometimes they take short cuts when writing tests.
- Some tests can be very difficult to write incrementally.
- It difficult to judge the completeness of a set of tests.

Pair Programming

- Programmers work in pairs.
- 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 this.
- Measurements suggest that development productivity with pair programming is similar to that of two people working independently.

Agile Project Management

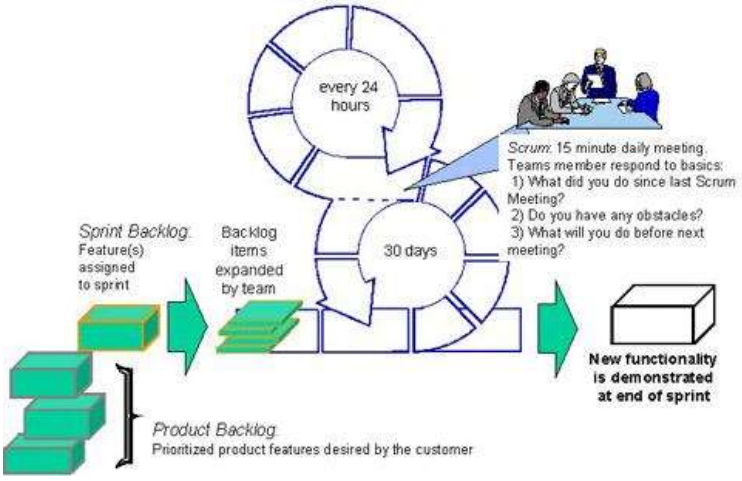
- The standard approach to project management is plan-driven.
- Agile project management requires a different approach, which is adapted to incremental development and the particular strengths of agile methods.

A project management technique that is effective when used with an agile process model.

SCRUM

The Scrum process

Scrum



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

OTHER AGILE MODELS

Adaptive Software Development

- Speculation
  - Adaptive cycle planning is used to define the set of software increments
- Collaboration
  - Requirements gathering techniques that involves team members, clients, and end-users
- Learning
  - Focus groups – clients and end-users
  - Formal technical reviews
  - Postmortems

Dynamic Systems Development Method

- Somewhat similar to RAD
- Pareto Principle: 80% of the software can be delivered in 20% of the time
- DSDM Life Cycle
  - Feasibility study
  - Business study
  - Functional model iteration
  - Design and build iteration
  - Implementation

Feature Driven Development

- Feature: “a client-valued function that can be implemented in two weeks or less.”
- 5 framework activities
  - Develop an overall model
  - Build a features list
  - Plan by feature
  - Design by feature
  - Build by feature