Software Engineering

Week # 5

LECTURER: ABDULRAHMAN QAIM

Testing in XP

Testing is central to XP and XP has developed an approach where the program is tested after every change has been made.

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.

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.

Customer involvement

The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system.

The customer who is part of the team writes tests as development proceeds. All new code is therefore validated to ensure that it is what the customer needs.

However, people adopting the customer role have limited time available and so cannot work full-time with the development team. They may feel that providing the requirements was enough of a contribution and so may be reluctant to get involved in the testing process.

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

- These testing components should be stand-alone, should simulate the submission of input to be tested and should check that the result meets the output specification. An automated test framework (e.g. Junit) is a system that makes it easy to write executable tests and submit a set of tests for execution.

As testing is automated, there is always a set of tests that can be quickly and easily executed

- Whenever any functionality is added to the system, the tests can be run and problems that the new code has introduced can be caught immediately.

XP testing difficulties

Programmers prefer programming to testing and sometimes they take short cuts when writing tests. For example, they may write incomplete tests that do not check for all possible exceptions that may occur.

Some tests can be very difficult to write incrementally. For example, in a complex user interface, it is often difficult to write unit tests for the code that implements the ‘display logic’ and workflow between screens.

It difficult to judge the completeness of a set of tests. Although you may have a lot of system tests, your test set may not provide complete coverage.

Pair programming

In pair programming, programmers sit together at the same workstation to develop the software.

Pairs are created dynamically so that all team members work with each other during the development process.

The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave.

Pair programming is not necessarily inefficient and there is evidence that a pair working together is more efficient than 2 programmers working separately.

Advantages of pair programming

It supports the idea of collective ownership and responsibility for the system.

- Individuals are not held responsible for problems with the code. Instead, the team has collective responsibility for resolving these problems.

It acts as an informal review process because each line of code is looked at by at least two people.

It helps support refactoring, which is a process of software improvement.

- Where pair programming and collective ownership are used, others benefit immediately from the refactoring so they are likely to support the process.

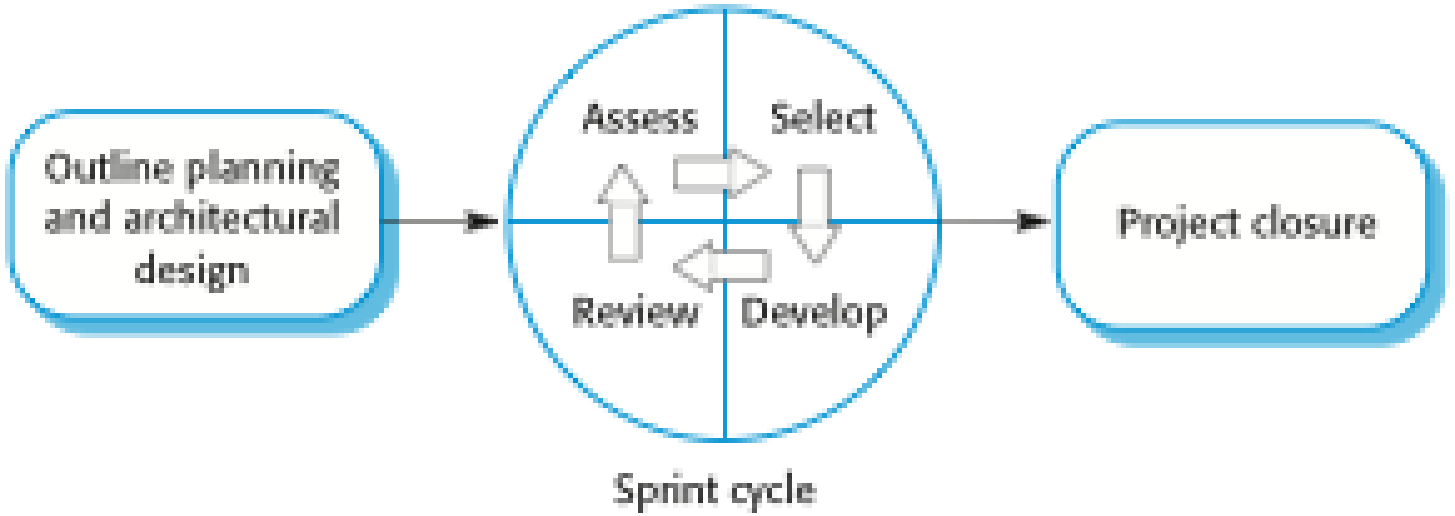
Scrum

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.

The Scrum process



The Sprint cycle

Sprints are fixed length, normally 2–4 weeks. They correspond to the development of a release of the system in XP.

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.

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 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.

Large systems development

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.

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.

Scaling up to large systems

For large systems development, it is not possible to focus only on the code of the system. You need to do more up-front design and system documentation

Cross-team communication mechanisms have to be designed and used. This should involve regular phone and video conferences between team members and frequent, short electronic meetings where teams update each other on progress.

Continuous integration, where the whole system is built every time any developer checks in a change, is practically impossible. However, it is essential to maintain frequent system builds and regular releases of the system.

Scaling out to large companies

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

A particular strength of extreme programming is the development of automated tests before a program feature is created. All tests must successfully execute when an increment is integrated into a system.

The Scrum method 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.

Scaling agile methods for large systems is difficult. Large systems need up-front design and some documentation.