Chapter 3: Agile Software Development

Your name:

Answer all questions. 1 mark per question

1. What are the shared characteristics of different approaches to rapid software development?

   The processes of specification, design and implementation are inter-leaved.

   The system is developed and delivered as a series of versions.

   User interfaces are often developed using an interactive development system that supports rapid UI development.

2. For what types of system are agile approaches to development particularly likely to be successful?

   Small and medium-sized software product development.

   Custom software development in an organization where there is a clear commitment from customers to become involved in the development process.

3. List 4 questions that should be asked when deciding whether or not to adopt an agile method of software development.

   Any 4 from those below. Others are also possible (see Ch 3)

   Is an incremental delivery strategy realistic?

   What type of system is being developed?

   What is the expected system lifetime?

   How is the development team organized?

   Is the system subject to external regulation?

   How large is the system that is being developed?

5. What are three important characteristics of extreme programming?

   Requirements expressed as scenarios,

   Pair programming,

   Test-first development

6. What is test-first development?

When a system feature is identified, the tests of the code implementing that feature are written before the code. Tests are automated and all tests are run when a new increment is added to the system.

7. What are the possible problems of test-first development?

Programmers may take short-cuts when developing tests so that the system tests are incomplete.

Some tests can be difficult to write incrementally.

It is difficult to estimate the completeness of a test set.

8. Briefly describe the advantages of pair programming.

It supports the idea of common ownership and responsibility for the code.

It serves as an informal code review process.

It helps support refactoring

Pair working
Names:

1. Explain the principles of agile methods in the way you are understanding, and give examples.

Hints: Students should not copy exactly the following descriptions.

1. Incremental development is supported through small, frequent releases of the system. Requirements are based on simple customer stories or scenarios that are used as a basis for deciding what functionality should be included in a system increment.

E.g.: some simple / basic functions are released first, and modified more then.

2. Customer involvement is supported through the continuous engagement of the customer in the development team. The customer representative takes part in the development and is responsible for defining acceptance tests for the system.

E.g.: some customers are involved in the development team to give feedback regularly.

3. People, not process, are supported through pair programming, collective ownership of the system code, and a sustainable development process that does not involve excessively long working hours.

E.g.: In the team, person A who is good at HTML is assigned to jobs of designing user interface pages, person B who is good at Java will do coding the back-end part...

4. Change is embraced through regular system releases to customers, test-first development, refactoring to avoid code degeneration, and continuous integration of new functionality.

E.g.: create some test cases first, then implement functions for the test cases. Functions are implemented incrementally.

5. Maintaining simplicity is supported by constant refactoring that improves code quality and by using simple designs that do not unnecessarily anticipate future changes to the system.

E.g.: implement simple functions enough for use

2. Discuss the following extreme programming practices, and give examples for each.

- **Pair programming**

  Developers work in pairs, checking each other's work and providing the support to always do a good job.

  E.g.: one member codes and the other one supports

- **Collective ownership**

  The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.

  E.g.: whenever a member codes a function, others can see and modify.

- **Continuous integration**

  As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.

  E.g.: when a member finishes a task/function, it will be put into the system.

- **Sustainable pace**

  Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity

  E.g.: leave the advanced functionalities for later versions to deliver the product on time.

- **On-site customer**

  A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

  E.g.: a member of the product development team is a customer who will provide requirements directly.