Hyperiondev

TASK

Agile Development

Visit our website

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

Introduction

Welcome to the Agile Development Task!

If you have ever spent any time around engineers, you have probably heard the words “Agile” bandied about quite a bit. In recent years “Agile” has become a buzzword in the business world, but it is not a new concept when it comes to software development. So what exactly is agile development? This task aims to answer that question as well as discuss one of the most popular agile methodologies, Extreme Programming.

Get in touch

Connect for support

Remember that with our courses, you’re not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

Copyright © 2018 Hyperion Development. All rights reserved.

What is Agile?

Today, software is part of almost all business operations. This software needs to be developed quickly to take advantage of new opportunities and to respond to competitive pressure in a rapidly changing business environment. Therefore, the most critical requirement for software system nowadays is rapid development and delivery. Some companies are even happy to trade software quality and compromise on requirements to achieve faster deployment of the software.

Agile methods are incremental development methods in which changes are made in small increments. With agile methods, new releases of the system are created and made available to customers every two or three weeks. They involve customers in the development process to get rapid feedback on changing requirements. Agile methods also minimise documentation by using informal communications rather than formal meetings and written documents.

Agile methods allow the development team to focus on the software rather than on its design and documentation. They are best suited to application development where the system requirements usually change rapidly during the development process. Using agile methods allows you to deliver working software to customers quickly. These customers can then suggest new requirements that can be added or suggest how to change requirements.

Agile development is a term that is used to refer to several different iterative and incremental software development methodologies. Some of the most popular agile methodologies are:

- Extreme Programming (XP)
- Scrum
- Crystal
- Dynamic Systems
- Development Method (DSDM)
- Lean Development
- Feature-Driven Development (FDD)

The best known of these methodologies is Extreme Programming. We will describe this methodology in greater detail later on in this task.

All of these agile methodologies share common values and principles which is derived from the Agile Manifesto found below:

Copyright © 2018 Hyperion Development. All rights reserved.

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 methodologies have worked well and were successful for some types of system development such as:

- Product development where a software company is developing a small or medium-sized product for sale

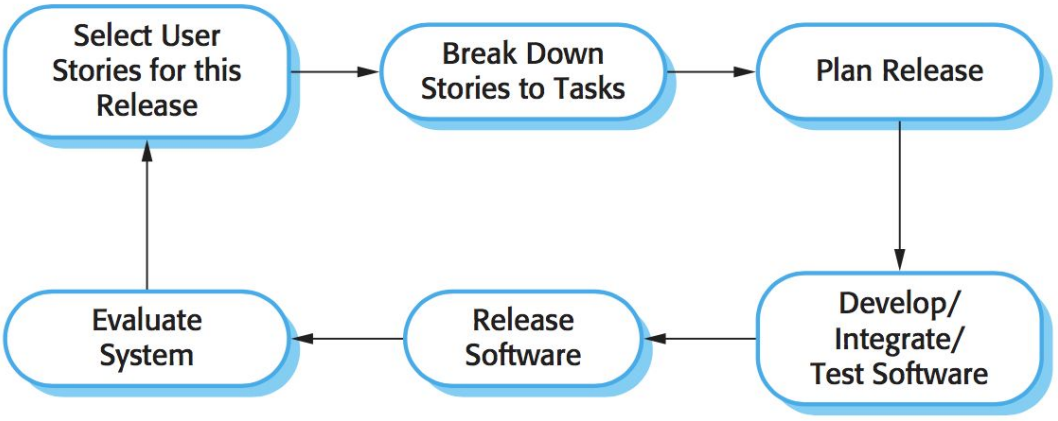
- Custom system development within an organisation, where customer is keen to be involved in the development process and where there are not a lot of external rules and regulations that affect the software.

Extreme Programming

One of the best known and widely used agile methodologies is Extreme Programming (XP). It is called “Extreme Programming” because this methodology pushes recognised good practice, such as iterative development, to “extreme” levels. For example, in a single day, several new versions of a system can be developed, integrated and tested, with XP.

XP expresses requirements as **user stories** which can be thought of as scenarios. These user stories are implemented directly as a series of tasks. A pair of programmers develop **tests** for each task before any code is written. When new code is integrated into the system, all these tests must be executed successfully. There is a short time gap between the release of each new version of the system.

Copyright © 2018 Hyperion Development. All rights reserved.



The XP release cycle (Sommerville 2010)

The diagram above shows the XP process to produce an increment (version) of the system that is being developed.

Like all other agile methods, Extreme Programming involves several practices which reflect the principles of the Agile Manifesto. We will discuss each of these practices in more detail below:

1. **Incremental development:** The system is released in frequent small intervals. Requirements are based on simple customer stories or scenarios which are recorded on Story Cards and used as a basis for deciding what functionality to be included in the system increment.

2. **Customer involvement:** The customer is constantly engaged in the development team. A customer representative takes part in the development and is responsible for defining the acceptance tests for the system.

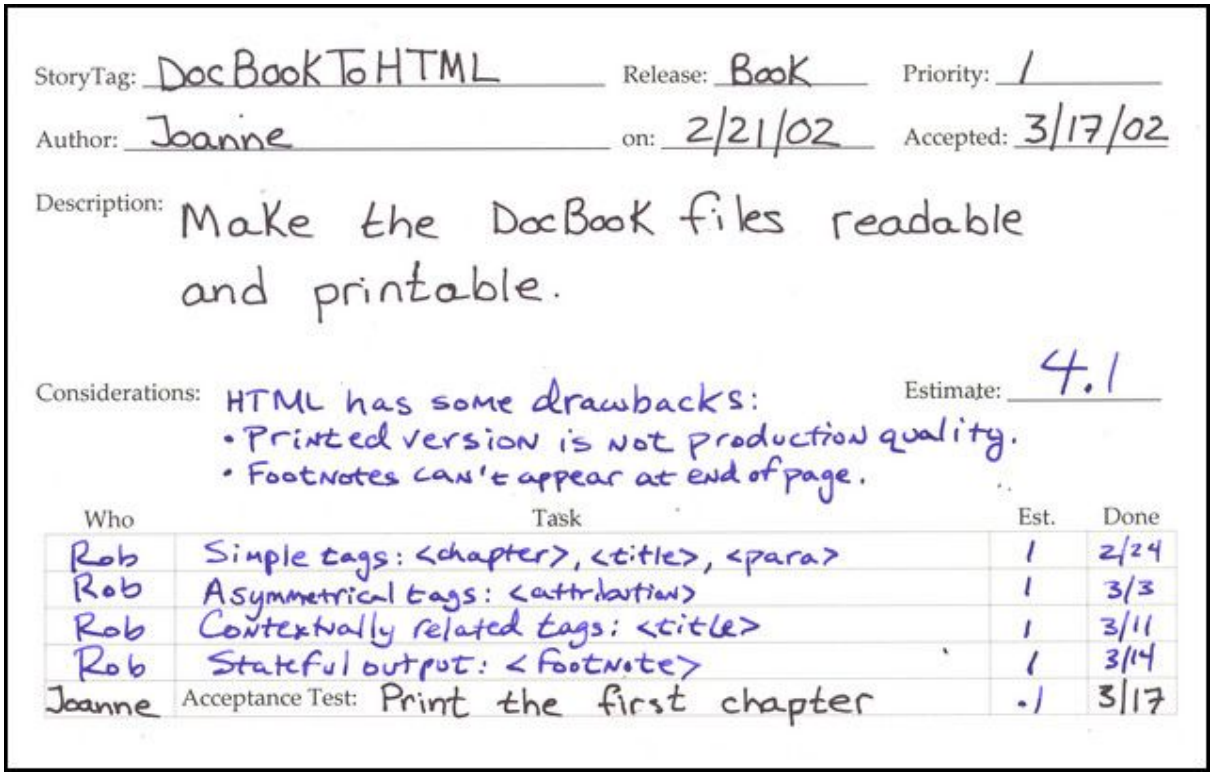
3. **People, not process:** Programming is done in pairs and the entire development team takes collective ownership of the system code. The development process also does not involve excessively long working hours.

4. **Change is embraced:** This is done through regularly releasing the system to customers and designing tests before code is written. The code is also refactored as soon as possible code improvements are found. Refactoring is the process of restructuring existing computer code to avoid code degeneration. New functionality is also constantly integrated.

Copyright © 2018 Hyperion Development. All rights reserved.

5. **Maintaining simplicity:** The constant refactoring of code improves code quality and simple designs are used that do not unnecessarily anticipate future changes to the system.

With an XP process customers are very much involved in specifying as well as prioritising system requirements. The customer is part of the development team and discusses scenarios with the other members of the team. All team members then develop a "**story card**" which is a short description of the customer's needs. The team then implements the scenario in a future version of the software.



StoryTag: DocBookToHTML Release: Book Priority: 1

Author: Joanne on: 2/21/02 Accepted: 3/17/02

Description: Make the DocBook files readable and printable.

Considerations: HTML has some drawbacks: Estimate: 4.1
- Printed version is not production quality.
- Footnotes can't appear at end of page.

Who	Task	Est.	Done
Rob	Simple tags: <chapter>, <title>, <para>	1	2/24
Rob	Asymmetrical tags: <attribution>	1	3/3
Rob	Contextually related tags: <title>	1	3/11
Rob	Stateful output: <footNote>	1	3/14
Joanne	Acceptance Test: Print the first chapter	.1	3/17

Example of a Story Card

Once the story cards have been developed, the development team breaks these down into tasks and estimates the effort and resources required for implementing each task. The customer is usually consulted to refine the requirements and prioritise the stories for implementation. Stories that can be used immediately to deliver useful business support should be implemented first. If new changes come to light, new story cards need to be developed. The customer needs to decide whether these changes should have priority over new functionality.

A "**spike**" is an increment in which no programming is done. This can be due to the team carrying out prototyping or trial development to better understand a problem and try to find a solution. A "spike" may also occur if the team needs to design the system architecture or to develop system documentation.

Copyright © 2018 Hyperion Development. All rights reserved.

With Extreme Programming, new versions of the software are delivered to customers roughly every two weeks and several new versions of the software can be built in a single day. The deadlines for releases are never missed. If the development teams experience and problems in the process, the customer is immediately consulted and some functionality is removed from the upcoming release.

Testing in XP

XP places a lot of emphasis on the importance of program testing. The XP approach to testing reduces the chances of introducing errors that were not discovered into the current version of the system.

In XP, the key features of testing are:

1. Test-first development
2. Incremental test development from scenarios
3. User involvement in the test development and validation
4. The use of automated testing frameworks

This is one of the most important innovations in XP. You write the tests before you write the code, rather than writing some code and then writing tests for that code. By doing this, you can run the test as the code is being written and discover problems during development.

In test-first development, team members assigned to a task, have to thoroughly understand the specification so that they are able to write tests for the system. This means that any omissions or ambiguities in the specification have to be clarified before implementation begins. It also avoids “test-lag” which happens when the developer works faster than the tester. When this happens, the implementation of the system gets further and further ahead without testing. Tests are often skipped to keep the project on schedule.

The development team assess each scenario that expresses the user requirements and breaks it down into tasks. Each task generates one or more unit tests that check the implementation described in that task.

Copyright © 2018 Hyperion Development. All rights reserved.

The customer helps to develop acceptance tests for scenarios that are to be implemented in the next release of the system. With acceptance testing, the system is tested using customer data to check that it meets the customer's real needs. Like development, acceptance testing in XP is incremental. As the development proceeds the customer, who is part of the team, writes tests.

For test-first development, test automation is essential. Tests are written as executable components that: should be standalone, should simulate the submission of input to be tested, and should check that the result meets the output specification before the task is implemented. An automated test framework makes it easy to write executable tests and submit a set of tests for execution. Since testing is automated, there is always a set of tests that can be quickly and easily executed.

Pair Programming

One of the most important practises introduced in XP is pair programming. With pair programming, programmers work in pairs to develop software. Both programmers sit together at the same workstation and work to develop the software. However, the same pairs don't always program together. All team members eventually will end up working with each other.

Some of the advantages of pair programming are:

1. It supports the idea of collective ownership and responsibility for the system. When the whole team owns the software, individuals are not held responsible for problems with the code. The entire team has collective responsibility for resolving these problems instead.

2. It acts as a type of informal review process since each line of code is looked at by at least two people. Even though pair programming probably doesn't find as many errors, is much cheaper inspection process than formal program inspections.

3. It helps support refactoring, which is a process of software improvement. With pair programming and collective ownership, others benefit immediately from the refactoring. Thus, they are likely to support the process.

Copyright © 2018 Hyperion Development. All rights reserved.

Compulsory Task 1

- Create a file called answers.txt
- Answer the following questions in answers.txt:
- Reread “Agile Manifesto”. Can you think of a situation in which one or more of the four “values” could get a software team into trouble?
- Describe agility in your own words.
- Why do you think requirements change so much?
- Explain why test-first development helps the programmer to develop a better understanding of the system requirements.

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we’ve done a good job?

Click here to share your thoughts anonymously.

Copyright © 2018 Hyperion Development. All rights reserved.