Hyperiondev

TASK

Agile Development

Visit our website

Introduction

WELCOME TO THE AGILE DEVELOPMENT TASK!

If you have ever spent any time around Engineers, you have probably heard the word “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!

WHAT IS AGILE?

When you think of the term “agile”, what comes to mind? A gymnast? A cheetah? What do these things have in common? Both the gymnast and the cheetah are able to move quickly and skillfully with seemingly little effort. This is exactly the basis for agile development: in a world where everything is changing so quickly, agile development is the only way for developers to keep up with the ever-changing demands.

Of the models that we explored in the *Software Process* task, I’m sure you can guess that the Incremental Development Model is the most agile. It allows developers to release programs quickly and make changes easily and cost-effectively. It also allows the developers to work closely with the clients to ensure feedback and subsequent changes are swift.

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 that are derived from the Agile Manifesto found below:

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

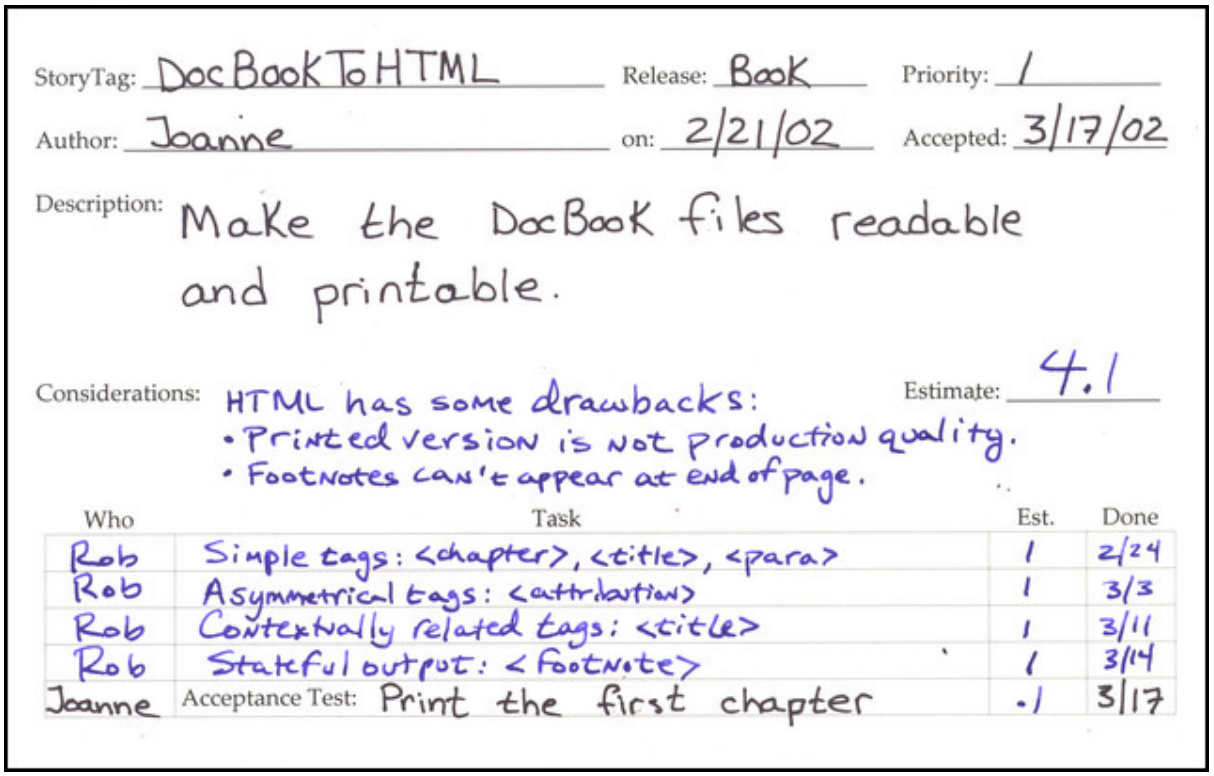
That is, while there is value in the items on the right, we value the items on the left more.
https://agilemanifesto.org/

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 (Beck, 2000 as cited in Sommerville, 2011, p. 64). For example, in a single day, several new versions of a system can be developed, integrated and tested, with XP.

XP takes a client's requirements for an application, known as *user stories*, and writes them onto a *story card*. The team then breaks down the requirement into smaller tasks. The tasks are then allocated to a pair of programmers who begin by developing tests for each task. Once the tests are created, the programmers can begin tackling their allocated tasks. Because there are multiple pairs of developers all working on different tasks concurrently, several versions of a programme could be created in a day. However, new versions are only released to clients every 2-3 weeks. This release deadline is never missed; if it is looking like there is too much that needs to be done before the deadline, it is the client's responsibility to decide what functionalities can be postponed to a later release.

STORY CARD



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

As mentioned above, a story card is written up for each new feature that the client requires the developing programme to have. When drawing up a story card, the requirement is broken down into tasks. Each task is given an estimated effort level and resources required. It is usually up to the client to decide which story cards are the most urgent and which should be released with the latest version.

PAIR PROGRAMMING

This is one of the key components of XP. As mentioned, developers work in pairs to complete tasks. Team members usually rotate so that all team members will have worked together at some point. One of the main benefits of this process is its promotion of refactoring: if your partner can't understand your code, how can you expect other people to? Similarly, having two people work on the same code creates an incidental code review system because there is always at least one other person who has seen a piece of code. Finally, having multiple people work on the same task creates a sense of collective ownership — every team member is responsible for the end product. This ensures a high-quality product. According to Cockburn and Williams (2000), pair programming causes the programming process to take 15% longer, but it had 15% fewer defects, which is quite significant. Programmers also reported a rise in job satisfaction through the collaboration.

TESTING IN XP

The testing process in XP is somewhat unusual, but there is a good reason for it. Unit tests for tasks are developed before the task itself is even started! Why? Firstly, this forces the developers to have an in-depth understanding of the specifications — you can't predict what can go wrong until you have a comprehensive understanding of what the task is meant to accomplish. Secondly, it checks that the implementation of the task is in line with the requirements given. Thirdly, it avoids test lag: when a developer works faster than the test creator. Finally, it prevents the introduction of errors that may have slipped through the cracks in the current version.

Tests are created using automated testing frameworks. These tests need to be standalone, executable components that can simulate submission input and should be able to check that the subsequent output is what is expected.

Clients help to create incremental acceptance tests for the release of the latest version. This includes testing the programme with real data to check if it meets the real needs of the client.

LINKING XP TO THE MANIFESTO

Looking at the information above, you should be able to see quite clearly how XP aligns with the Agile Manifesto, namely:

1. **Individuals and interactions over processes and tools:** Programming is done in pairs and the whole development team is accountable for the code. The development process also does not involve excessively long working hours.

2. **Working software over comprehensive documentation:** The system is released every few weeks. Requirements are based on user stories recorded on story cards. These guide what features are to be included in the next version of the program. The constant refactoring of code improves code quality and simple designs are used.

3. **Customer collaboration over contract negotiation:** The client is a key member of the development team and is responsible for defining the acceptance tests for the system.

4. **Responding to change over following a plan:** This is done through regular releases and designing tests before code is written. The code is also refactored by the pair programmers often. New functionality is also constantly integrated.

Compulsory Task

- 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

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

Think that 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.

References:

Cockburn, Alistair & Williams, Laurie. (2000). *The Costs and Benefits of Pair Programming*.

Sommerville, I. (2011). *Software Engineering 9*. 9th ed. Boston: Pearson Education, Inc., pp. 56-81.