

**Department of Computing**

**Te Horo Rorohiko**

**CH3880 Bachelor of Information and Communications Technology**

**BCCE301 Cooperative Education Project**

**Semester 2, 2017**

**JavaScript Online Quiz for Moodle3**

**Ara Institute of Canterbury**

**METHODOLOGY ESSAY**

Daming Zhang

99137058

Word Count:

### 

# Part A - Theory – Agile Project Management

## Introduction

This document is part of the JavaScript Online Quiz for Moodle3 project for the Graduate Diploma of ICT at Ara Institute of Canterbury. In this essay, the methodology that I used in the project is Extreme Programing planning method. The report is separated into four sections.

## History

Agile management of IT project is widely used for software development. In the earlier stages of development the norm was to use “heavy” methodologies such as Waterfall or Spiral for software development. As a common agile method, the first Extreme Programming project was started March 6, 1996. Extreme Programming is one of several popular Agile Processes. Extreme Programing was developed in the mid 1990’s, but at that time there were not yet any guidelines that detailed what Agile was (Wells, D. , n.d., para. 1) . In 2001 seventeen software developers meet in Utah to discuss the new ways of managing software projects and published the Agile Manifesto for Software Development.(The Agile Alliance). The Manifesto underlined common values and principals of all agile projects and led to what Agile is today. Agile Project Management is a lightweight method for software development projects.

## What is Extreme Programing(XP)?

Extreme Programing has been applied to be very successful at many companies of different sizes and industries all over the world. It is successful method because it leads customer to fell satisfied. It is not easy to deliver everything that as developers planed forego on some date very far in the future. With using this method, the process delivers the achievement as you plan it. Extreme Programming makes developers to respond confidently to changing requirements or adapting to client feedback.

The Extreme Programing emphasizes teamwork. Clients, managers, and developers are all equal important role in the collaborative team. It implements a simple and effective environment that encourages and enables developing teams to become highly productive. The team organizes itself about the problem and solve it very efficiently.

There are five essential ways that Extreme Programming can improve a software project: communication, feedback, simplicity, courage, and respect.

Programmers communicate with their customers, clients, managers and other programmers constantly. They deliver the production to the clients as early as they can and implement the changed requirements as the clients suggested. By every small unique contribution of every team member, the Extreme Programmers respond to changing requirements.

## Why Extreme Programing(XP)?

When a company wants to full the extreme program in the transition and make the use of program beyond the project development team, it needs to face some problems.

1. A bad communication between clients and developers.
2. Misunderstanding of user stories, the plan, and velocity.
3. A blurry deliver credible plans and schedules.

So be agile in adapting to client feedback and changed requirements is very important for software management.

## Advantages and disadvantages

### Advantages:

1. The first advantage of this method is the cost estimates are based on software features instead of developer activity. It allows clients to make a decision on what are the priority needs and what are the exclusive needs depending on the time consuming and budget. By selecting the important requirement first, the client gets the highest value with the least time or budget spent, and this can trade off on the tiny increase in the project with the cost on additional features. The approach also allows both the user and the client to "pull the plug" on development during developing period at any time and the functional code still keeps highly valuable, even if it is incomplete.

2. The second advantage of the approach is resilience. In realistic life, requirements keep changing all the time either because of emerging new business features or simply because the original requirements was not complete. Extreme Programming plans and organise the changed requirements through user stories at the beginning of iterations, and get feedback during the time of iterations.

3. The third advantage is Cost Savings. Extreme Programming cut off unproductive activities to lower costs. It makes project developers to focus on coding instead of wasting time on insignificance work and meetings. The time spend on making changes increases software budgets. The traditional programming approaches make changes based on client’s feedback at the end of the delivery lifecycle compare to Extreme Programming that it allows changes at the development phase.

4. With using this method, it reduces the risks for the project. Extreme Programming reduces the risks that are related to programming. Conventional programming depends a lot on the key members or important members in a team. By breaking the all tasks into several modules, it separates the risks and reduces the dependence on any one in the team, no matter how critical the member is in the team.

5. Extreme Programming also make Satisfaction to employee, as the essay mentioned, it reduces the importance of individuals during the development time of process and gives every member satisfaction and retention. Extreme Programming is a value-led method that with just little scope for overtime, it can sets fixed worktime. Developers and managers constantly communicate to clients and gets feedback from them to ensure the delivery to be finished before the deadline.

### Disadvantages:

The biggest disadvantage of Extreme Programming is that it constantly needs the involvement of the client. Basically, developers need to communicate to client periodic. They need several times of meeting to decide every changes of the project. The method itself success depends on data gathering at many phases of the developing process. Some clients may not be available, and many others may do like such a constant involvement. Extreme Programming code is an approach that focus on program-centered rather than a design-centered approach, and with the fewer proper documentation, it generates issues in the project when project developers leave and new developers come in later.

# Part B – Industrial Practice Observation

In this section, it focus on industrial practice observation. During the time I was developing the program, I gained the way of how to use the method to manage it appropriately and observed the EP development management process. It basically includes 11 steps:

#### Listening**[**[**edit**](https://en.wikipedia.org/w/index.php?title=Extreme_programming&action=edit&section=9)**]**

Programmers must listen to what the customers need the system to do, what "[business logic](https://en.wikipedia.org/wiki/Business_logic)" is needed. They must understand these needs well enough to give the customer feedback about the technical aspects of how the problem might be solved, or cannot be solved. Communication between the customer and programmer is further addressed in the [*planning game*](https://en.wikipedia.org/wiki/Planning_game).

#### Designing**[**[**edit**](https://en.wikipedia.org/w/index.php?title=Extreme_programming&action=edit&section=10)**]**

From the point of view of simplicity, of course one could say that system development doesn't need more than coding, testing and listening. If those activities are performed well, the result should always be a system that works. In practice, this will not work. One can come a long way without designing but at a given time one will get stuck. The system becomes too complex and the dependencies within the system cease to be clear. One can avoid this by creating a design structure that organizes the logic in the system. Good design will avoid lots of dependencies within a system; this means that changing one part of the system will not affect other parts of the system.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

#### Simplicity**[**[**edit**](https://en.wikipedia.org/w/index.php?title=Extreme_programming&action=edit&section=13)**]**

Extreme programming encourages starting with the simplest solution. Extra functionality can then be added later. The difference between this approach and more conventional system development methods is the focus on designing and coding for the needs of today instead of those of tomorrow, next week, or next month. This is sometimes summed up as the "[You aren't gonna need it](https://en.wikipedia.org/wiki/You_aren%27t_gonna_need_it)" (YAGNI) approach.[[8]](https://en.wikipedia.org/wiki/Extreme_programming#cite_note-tr-8) Proponents of XP acknowledge the disadvantage that this can sometimes entail more effort tomorrow to change the system; their claim is that this is more than compensated for by the advantage of not investing in possible future requirements that might change before they become relevant. Coding and designing for uncertain future requirements implies the risk of spending resources on something that might not be needed, while perhaps delaying crucial features. Related to the "communication" value, simplicity in design and coding should improve the quality of communication. A simple design with very simple code could be easily understood by most programmers in the team.

Release Plan

In the beginning of the project, I wrote a proposal that includes a roughly plan for the whole project. It had 4 activities. Base on the wide of my technical knowledge, I estimated the deadline for each activity and all the activities needed to be done in 12 weeks. Then I discussed with my client the roughly deadline time for each tasks. Also, the plan would probably be changed and the requirements would be renegotiated along the project proceeding.

Iteration Plan

During the time I was going to start to develop each activity, I made a more detailed plan for it. I separated the activity into several features. Each feature was one task. Then on the basis of difficulty of each feature I classify all those features as big tasks and small tasks. What is more, I discussed with my client to get to know what is my client’s priority among all the tasks. The most important task can still separate into several small tasks. Instead of delivering everything on some day far in the future, I could confidently finish the changing requirements with using Extreme Programming planning method.

Acceptance Test

Before each increment of project requirements, I planned and wrote tests. In order to shorten the total time of development, the purpose of these tests are to judge if it is acceptable for delivery and evaluate the system’s acceptance of the requirements. Normally, black box testing is used in Acceptance Testing. Usually, it does not follow a strict procedure. Some test documents have been developed in parallel while the software is developing.

Stand Up Meeting

Building software systems requires communicating system requirements to the developers of the system. In formal software development methodologies, this task is accomplished through documentation. Extreme programming techniques can be viewed as methods for rapidly building and disseminating institutional knowledge among members of a development team. The goal is to give all developers a shared view of the system which matches the view held by the users of the system. To this end, extreme programming favors simple designs, common metaphors, collaboration of users and programmers, frequent verbal communication, and feedback.

Pair Negotiation

* Feedback from the customer: The functional tests (aka [acceptance tests](https://en.wikipedia.org/wiki/Acceptance_tests)) are written by the customer and the testers. They will get concrete feedback about the current state of their system. This review is planned once in every two or three weeks so the customer can easily steer the development.
* Feedback from the team: When customers come up with new requirements in the planning game the team directly gives an estimation of the time that it will take to implement.

Unit Test:

* [Unit tests](https://en.wikipedia.org/wiki/Unit_test) determine whether a given feature works as intended. Programmers write as many automated tests as they can think of that might "break" the code; if all tests run successfully, then the coding is complete. Every piece of code that is written is tested before moving on to the next feature.

Pair Programming:

* Feedback from the system: by writing [unit tests](https://en.wikipedia.org/wiki/Unit_test),[[5]](https://en.wikipedia.org/wiki/Extreme_programming#cite_note-Cworld92-5) or running periodic integration tests, the programmers have direct feedback from the state of the system after implementing changes.

Coding

The advocates of XP argue that the only truly important product of the system development process is code – software instructions that a computer can interpret. Without code, there is no working product.

Coding can also be used to figure out the most suitable solution. Coding can also help to communicate thoughts about programming problems. A programmer dealing with a complex programming problem, or finding it hard to explain the solution to fellow programmers, might code it in a simplified manner and use the code to demonstrate what he or she means. Code, say the proponents of this position, is always clear and concise and cannot be interpreted in more than one way. Other programmers can give feedback on this code by also coding their thoughts.

# Part C – Actual Project Practices VS Theory

In this section, I am looking at what I have done during the development period, compare and contrast actual outcomes with the theory.

Product Owner & Client

First of all, there are three critical roles in the theory which are developers, managers and clients. In the project Anne was the client, my supervisor Mike and Phillip were the managers and I was the developer.

The theory says customer first create a release plan and choose the stories to implement.

Sprint Review

Compare to the reality, Anne told me the background and existed issues of the project. After that picked up 4 activities to let me implement and order the implementation by importance. Which meant what are the must do, could do. So I made a rough plan to give different length of time for each activity on the basis of the difficulty of each of them. In this way, the theory helped me to manage the project scheduled.

**Simplicity**

Simple solutions are cheaper and quicker to implement than complex solutions. Therefore, XP always tries to find the simplest solutions.

**Communication**

All team members should communicate intensively with each other . Through personal dialogue, misunderstandings can be eliminated very quickly; questions can likewise be answered very promptly. As a result, documentation can sometimes be waived.

**Courage:**To use these values and at the same time communicate openly requires a great deal of courage, especially for those project members who are not used to acting in accordance with these values. It takes courage to scale the requirements of the customer down to the most important points with continuous feedback and openness as well as direct communication with the customer and other project members.

Then base on the theory, developers need to implement each planned task in turn by the order that defined by the client. There is one different that developers are always more than one person, but in this project there is only one developer which is me. The project could be interrupted if I got sick.

Base on the theory, every big task can separate into several small tasks. So the iteration of planning ends when the time runs out or all the required tasks are finished. when the implementation is “finished”, the developers achieved milestone. If the developers have finished all the tasks before the end of the iteration, I could ask the client for a new requirement. It shortens the developing cycle.

The theory has mentioned that this method is very effective to develop projects which the requirement are changed regularly (Wells, D. , n.d., para. 3). In the industry practise, my project changed quite often. I worked 24 hours each week to ensure that I can finish all the required activities in 12 weeks. Since the requirements were always changed, so I followed the rules of Extreme Programming to organise the project. I separated all the features, then analysis the importance of each feature. What is more, I estimated time consuming of each feature base on the difficulty of implementing. At last I put the description, importance and time consuming for each feature. In the theory each of the feature with description, consuming is called a XP card. After doing this, I showed all the feature XP card to my client. Let her picked some of them that she really wanted to make them work. and the total number of time consuming needed to equal to 24 hours. The method really worked, it ensured that I always accomplish tasks earlier and did some more tasks. However, the XP card game seemed not quite necessary. There are two reasons, one reason was my client had experience with communicating to developers and she was quite familiar with the project, some other developers had done the similar project before, so she always picked up the appropriate tasks which were all essential and adjust total time consuming of all the task to around 24 hours. This was really good for me and make me effective to finish the project. However, it could happen that a client does not have too much experience, so developers need to explain more to client and negotiate to client with the new requirements. In my case, all the features had to be done otherwise the activity would not work, so me and my client were quite easy to meet the agreement.

At every moment, the design runs all the tests, communicates everything the programmers want to communicate, contains no duplicate code, and has the fewest possible classes and methods. This rule can be summarized as, 'Say everything once and only once.'

Conclusion

In conclusion, we can know what is suggested by Extreme Programming. Extreme Programming teams work in a strict priority order. Features to be developed are prioritized by the customer (Cohn, M. n.d., para 4) and the team is required to work on them in that order.

XP teams typically work in iterations

Business and development cooperate to produce the maximum business value as rapidly as possible. The planning game happens at various scales, but the basic rules are always the same:

* Business comes up with a list of desired features for the system in user story format.
* Development estimates how much effort each story will take, and how much effort the team can produce in a given time interval.
* Business then decides which stories to implement in what order.

Start with the smallest useful feature set. Release early and often, adding a few features each time.

the idea of the Extreme Programming (XP) approach now follows. XP has been chosen here as an example because it describes the most extreme case of agile procedure methodology.

Reference:

Extreme programming. (2017, October 20). Retrieved October 30, 2017, from https://en.wikipedia.org/wiki/Extreme\_programming

Wells, D. (n.d.). Extreme Programming: A Gentle Introduction. Retrieved October 31, 2017, from http://www.extremeprogramming.org/

Cohn, M. (n.d.). Differences Between Scrum and Extreme Programming. Retrieved November 08, 2017, from https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming

AG, I. (n.d.). Extreme Programming (XP) | Scrum Compact. Retrieved November 08, 2017, from https://www.itemis.com/en/agile/scrum/compact/fundamentals-of-project-management/extreme-programming-xp