|  |  |
| --- | --- |
| P:\Games Programming Degree\Implelmentation\The_Academy_of_Interactive_Entertainment_aie.jpg | CIT_CMYK2 |

**INFT569 Project Architecture 3**

**Assessment Task 1**

**Student Name: xxxxx yyyyyy**

**Student Number: CIT999999**

**Date Due: DD MMM 2099**

Table of Contents

[Glossary of Terms 3](#_Toc252439167)

[Executive Summary 4](#_Toc252439168)

[Introduction or Background 5](#_Toc252439169)

[Discussion 5](#_Toc252439170)

[Subheading One 5](#_Toc252439171)

[Subheading Two 5](#_Toc252439172)

[Conclusion 5](#_Toc252439173)

[Bibliography 6](#_Toc252439174)

# Glossary of Terms

|  |  |
| --- | --- |
| ACT | Australian Capital Territory |
| AIE | Academy of Interactive Entertainment |
|  |  |
|  |  |
|  | Entries must be in alphabetical order. |
|  |  |
|  |  |
| CIT | Canberra Institute of Technology |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Executive Summary

Usually two or three paragraphs, but seldom longer than one page. This is usually on a page of its own.

Testing about the things

# Introduction or Background

Most often used to justify and define your approach to the question. Your approach is often based on the research approaches of others. May include definitions of main terms used in framing your approach. Be sure to reference all sources.

# Discussion

This is where you present your main discussion. All of your information must reference its source, and the source must be in the bibliography at the end.

## Subheading One

You may use subheadings to segment the discussion.

## Subheading Two

This approach provides a focus and helps tie the discussion points together.

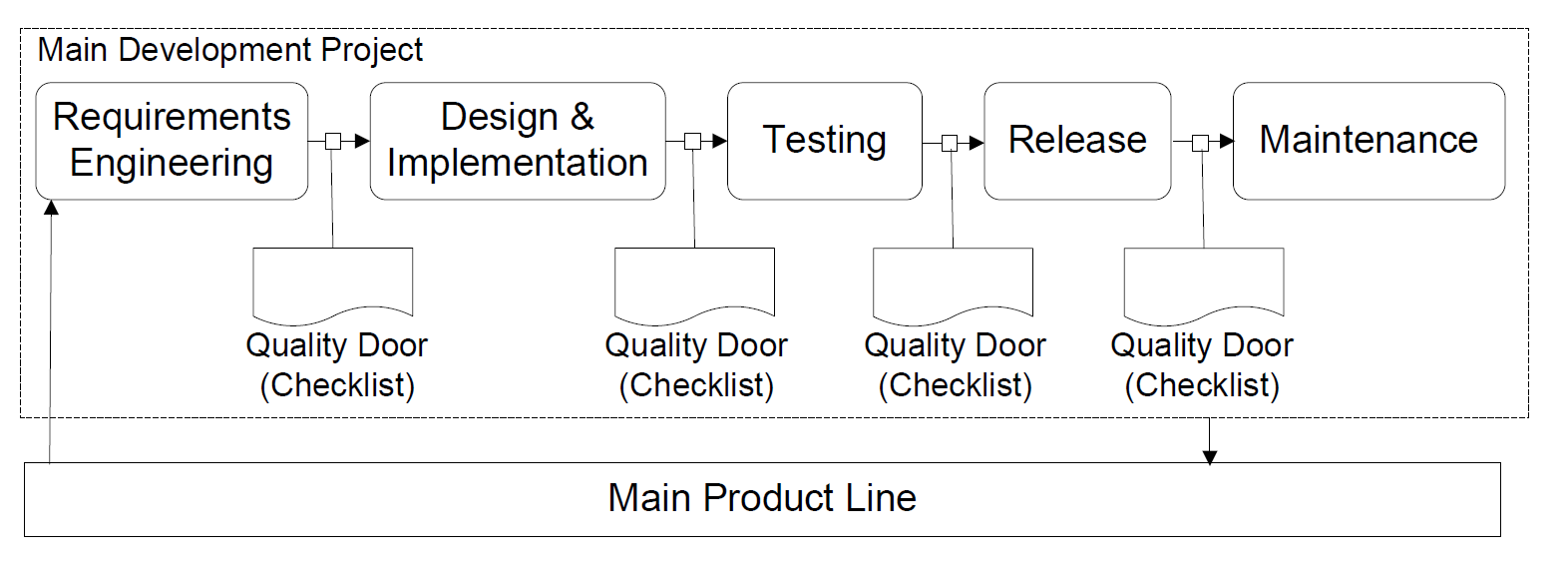
# Conclusion

This is where you tie together all the points you have made. You would not normally be introducing new material at this point, so don’t be surprised if there are no references under this heading.

# Bibliography

Entries under this heading are in alphabetical order using the approved format.

# The Waterfall Model



## The Methodology of the Waterfall Model

### Requirements Phase

**Requirements In Software Definition:** The capabilities that a system must have **(functional)** or properties of that system **(non-functional)** that meet the users' needs to perform a specific set of tasks (within a defined scope).

**Example of Functional requirements**: Print, Save, Shoot.

**Example of Non-functional requirements**: How easy is it to use, performance and security, GUI Design?

**Goal of this phase** Identify and Document all Functional and Non-functional requirements

**Process Requirements phase**

The first step of this phase is too identified and document the customers need in a high level that what requirements they need , after that in second step the requirements are refined so they can be used to design and implementation. The third step, the requirements are stored in a requirements repository or requirements database. From this repository or database we will select the things that we need. The requirements selected depends on the available resources for this project. Since a new product will not be built on scratch, parts from old product are used as input to this step as well. The forth step and the last step of this phase, Quality gate, the goal of quality gate is too first check whether all requirements are understood, secondly are the requirements agreed upon and documented. Lastly is to check whether the stakeholders are identified and also whether the solution that plan out would match the business strategy.

### Design Phase and Implementation

**System** **Design In Software Definition:**

System design is the process of defining the components, modules, interfaces, and data for a system (which can be done by UML) to satisfy specified requirements. System development is the process of creating or altering systems, along with the processes, practices, models, and methodologies used to develop them.

**Goal of the Design phase**

Define all the components, modules, interfaces, and data for a system that can meet your business needs.

**Process of Design Phase**

The design phase is further categorized into two sections, i.e. system design and component design. The system design contains details and specifications of the whole system and explains how each component of the system will interact with others. The component design contains specifications as to how each component will work separately and how results from one component will travel to another.

**Process of Implementation Phase**

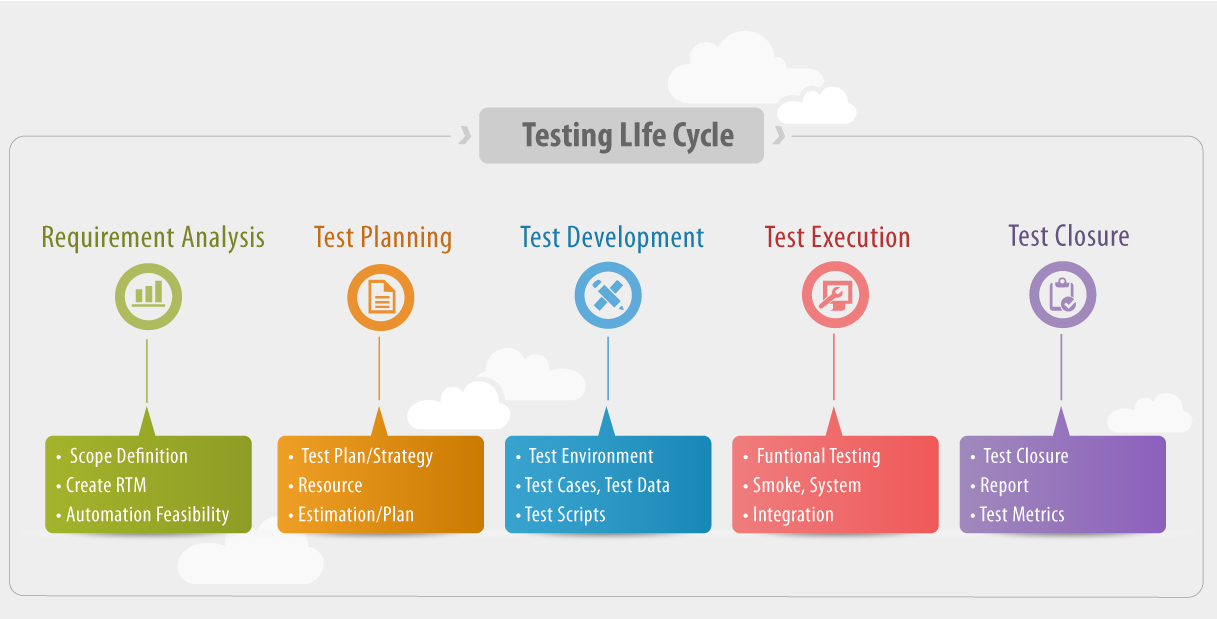
After the implementation phase the developers would need to also conduct basic unit testing before handing the developed code to the test phase. The quality gate checklist verifies whether the architecture has been evaluated, whether there are deviation from the requirements compared to the previous quality gate decision, and whether from the requirements compared to the previous gate decision, and whether there is a deviation from planned time-line, effort, or product scope.

### Testing Phase

**Testing In Software Definition**

Mechanism to assure quality of a product, system, or capability (e.g., right product, built right). To be effective, testing cannot occur only at the end of a development. It must be addressed continuously throughout the entire life cycle.

**Process of testing phase**

****

**Requirement Analysis:**

In this phase we would need toidentified and document in a high level that what should be tested

**Testing Planning:**

In this phase a Test Plan we be create to provide what test would be tested, the test documentation should provide information of what type, level and methods of testing

The following is a list of type, level and methods of testing

* [Testing methods](https://en.wikipedia.org/wiki/Software_testing#Testing_methods)
  + [3.1 Static vs. dynamic testing](https://en.wikipedia.org/wiki/Software_testing#Static_vs._dynamic_testing)
  + [3.2 The box approach](https://en.wikipedia.org/wiki/Software_testing#The_box_approach)
    - [3.2.1 White-box testing](https://en.wikipedia.org/wiki/Software_testing#White-box_testing)
    - [3.2.2 Black-box testing](https://en.wikipedia.org/wiki/Software_testing#Black-box_testing)
      * [3.2.2.1 Visual testing](https://en.wikipedia.org/wiki/Software_testing#Visual_testing)
    - [3.2.3 Grey-box testing](https://en.wikipedia.org/wiki/Software_testing#Grey-box_testing)
* [Testing levels](https://en.wikipedia.org/wiki/Software_testing#Testing_levels)
  + 4.1 Unit testing
  + [4.2 Integration testing](https://en.wikipedia.org/wiki/Software_testing#Integration_testing)
  + 4.3 Component interface testing
  + 4.4 System testing
  + 4.5 Operational Acceptance testing
* [Testing Types](https://en.wikipedia.org/wiki/Software_testing#Testing_Types)
  + [5.1 Installation testing](https://en.wikipedia.org/wiki/Software_testing#Installation_testing)
  + [5.2 Compatibility testing](https://en.wikipedia.org/wiki/Software_testing#Compatibility_testing)
  + [5.3 Smoke and sanity testing](https://en.wikipedia.org/wiki/Software_testing#Smoke_and_sanity_testing)
  + [5.4 Regression testing](https://en.wikipedia.org/wiki/Software_testing#Regression_testing)
  + [5.5 Acceptance testing](https://en.wikipedia.org/wiki/Software_testing#Acceptance_testing)
  + 5.6 Alpha testing
  + 5.7 Beta testing
  + [5.8 Functional vs non-functional testing](https://en.wikipedia.org/wiki/Software_testing#Functional_vs_non-functional_testing)
  + [5.9 Destructive testing](https://en.wikipedia.org/wiki/Software_testing#Destructive_testing)
  + [5.10 Software performance testing](https://en.wikipedia.org/wiki/Software_testing#Software_performance_testing)
  + [5.11 Usability testing](https://en.wikipedia.org/wiki/Software_testing#Usability_testing)
  + [5.12 Accessibility testing](https://en.wikipedia.org/wiki/Software_testing#Accessibility_testing)
  + [5.13 Security testing](https://en.wikipedia.org/wiki/Software_testing#Security_testing)
  + [5.14 Internationalization and localization](https://en.wikipedia.org/wiki/Software_testing#Internationalization_and_localization)
  + [5.15 Development testing](https://en.wikipedia.org/wiki/Software_testing#Development_testing)
  + [5.16 A/B testing](https://en.wikipedia.org/wiki/Software_testing#A.2FB_testing)
  + [5.17 Concurrent testing](https://en.wikipedia.org/wiki/Software_testing#Concurrent_testing)
  + [5.18 Conformance testing or type testing](https://en.wikipedia.org/wiki/Software_testing#Conformance_testing_or_type_testing)

**Testing Development:**

This step create test data for the test type, level and methods

**Test Execution:**

This step start the test and record all the test.

**Test Closure:**

In this step we would need to compare the result that we wanted and the actually result. Test Metrics is technical word that means the result that we wanted.

### Maintenance

After the product has been released to the customer it has to check did the customer discover any problems in the product, if yes. The customer will report the problem to the company and we should give support in solving them. If the problems are due to faults in the product, packages for updating the system are delivered to the customers.

3.2. The Strengths of the Waterfall Model ........................................... Error! Bookmark not defined. 3.2.1. Fixed Requirements............................................................... Error! Bookmark not defined. 3.2.2. Clear Phases........................................................................... Error! Bookmark not defined. 3.2.3. Focused Development ........................................................... Error! Bookmark not defined. 3.3. The Weaknesses of the Waterfall Model....................................... Error! Bookmark not defined. 3.3.1. Reduced Customer Feedback ................................................ Error! Bookmark not defined. 3.3.2. Inflexibility.............................................................................. Error! Bookmark not defined. 3.3.3. Increased Planning................................................................. Error! Bookmark not defined. 3.3.4. Requirements Control............................................................ Error! Bookmark not defined.

<https://books.google.com.au/books?id=CDWRq0B9e5kC&pg=PA40&lpg=PA40&dq=rad+model+AND+%22Business%22+AND+%22example%22&source=bl&ots=fuCUS_u2MY&sig=Q-YgELNPER-7GcfhJX66Ce1AuQk&hl=en&sa=X&ved=0CCYQ6AEwAjgKahUKEwjU2deEr6HHAhVnfaYKHZW5AEg#v=onepage&q=rad%20model%20AND%20%22Business%22%20AND%20%22example%22&f=false>

4. Rapid Application Development............................................................. Error! Bookmark not defined. 4.1. The Methodology of RAD............................................................... Error! Bookmark not defined. 4.2. The Strengths of RAD..................................................................... Error! Bookmark not defined. 4.2.1. Customer Involvement........................................................... Error! Bookmark not defined. 4.2.2. Rapid Development................................................................ Error! Bookmark not defined. 4.2.3. Scheduled Development........................................................ Error! Bookmark not defined. 4.3. The Weaknesses of RAD................................................................. Error! Bookmark not defined. 4.3.1. CASE Reliance......................................................................... Error! Bookmark not defined. 4.3.2. Customer Control................................................................... Error! Bookmark not defined. 4.3.3. Feature Loss........................................................................... Error! Bookmark not defined. Project Architecture 3 Page 4 of 14 Assessment Task 2 4.3.4. Customer Integration Requirement....................................... Error! Bookmark not defined. 4.3.5. Training .................................................................................. Error! Bookmark not defined.

# Scrum

## The Methodology of Scrum

### Product Backlog

Product Backlog is a card that include the features you wanted in this product also it contain a short description of all functionally desired in the product , the description usually includes the following format , As a ….. I want ……... Usually the product owner we begin writing down everything that is more than enough for the first sprint. Notice that the Scrum product backlog is allowed to grow and change as more is leaned about the product and its customers. The product backlog have different types of items including: Features, Bugs, Technical work and Knowledge acquisition.

### Sprint

You can imagine a sprint is a loop. Which inside the loop contains what work would be need to done in a set of period of time no matter how basic the product is. Every sprints begins with the sprint planning meeting, in which the product owner and the team discuss which stories will be move from the product backlog to the sprint backlog. The product owner will have responsibility of the product owner to determine what work the team will do, the product owner cannot add more or micromanage. The product can cancel a sprint. It only should happen it if the business needs changes.

### User Story

User story is a card that include information identifies who the end user is, what the end users wants. The model of the user story is most often written like: “As a [end user role], I want [the desire] so that [the rationale]”. The difference between product backlog and user story is the user story means “idea have not been proved” by the product owner ,which the product backlog is “ideas that have been proved” by the owner.

### Roles

Product Owner

The product owner is typically a project’s key stakeholder. Since the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team. So to be able the successfully starting any agile software development project the product owner would have reasonability to decide what should be in the product backlog and what shouldn’t be and also they have will select the amount of work they believe the team can do each sprint , and how much will be required. The large part of the product owner is to communicate. The product owner requires working closely with key stakeholders throughout the organization and beyond, so he or she must be able to communicate different messages to different people about the project at any given time.

Scrum Master

The responsibilities of the scrum master is making sure a scrum team lives by the value and practices of scrum. The scrum master is like a coach in the team try the push the team to their best performs. The scrum master also needs to know how it can create a balance between the project key stakeholder and the scrum team. To able to maximum the team performs. He or she would need to remove any impediments to process, facilitating meetings and doing things like working with the product owner to make sure the product backlog is in good shape and ready for the next sprint. Also the scrum master make sure that no team members overcommit themselves to what they can achieve during a sprint due to pressure from anoverly aggressive product owner. The scrum Master is commonly filled by a former project manager or a technical team leader but there are no limits of who should be the scrum master.

Scrum Team

A Scrum team in a [Scrum](https://www.mountaingoatsoftware.com/agile/scrum) environment does not include any of the traditional software engineering roles such as programmer, designer, tester or architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Because of this, Scrum teams develop a deep form of camaraderie and a feeling that "we're all in this together." A typical scrum team is usually from five to nine people. Rather than scaling by having a large team. Scrum recommend that projects scale through teams to teams. I n this way projects with more than 500 people and have consulted on projects with more than 1000.

## The Strengths of Scrum

Easy Adoption

Short Development Cycles

Reduced Planning

Increased Motivation

## The Weaknesses of Scrum

Integrated Testing

Organisation

# TDD



## The Methodology of TDD

Test-driven development is a programming technique where tests are written before the actual program code.

Characteristics of a Unit Test

## The Strengths of TDD

### Quality Improvement

### Fast Development

### High Code Coverage

### Encourages Separation of Concerns

### Rapid Responsiveness

The Weaknesses of TDD

### Little Management Guidance

### Difficult to Test Components

### Test Suite Speed

### Training Difficulties

### Test Development Time

<https://www.mountaingoatsoftware.com/agile/scrum/product-backlog/example>

<http://www.slideshare.net/MrSMAk/extreme-programming-12047889>

# Extreme Programming

## The Methodology of Extreme Programming

### The Extreme Programming Lifecycle

The XP consist of five phases which are: Exploration, Planning, Iterations to release,

Productionizing phase , Maintenance Phase and Death phase .

Exploration Phase

In this phase the customers would need to write out the story cards that they wish to be included in the first release. Each story card describes a feature to be added into the program. At the same time the project team would need familiarize themselves with the first tools, second technology and last the practices that they may use in this project. For the technology that they use will be tested and the architecture possibilities for the system are explored by building a prototype of the system. This phase usually take between a few weeks to few months, depending on how large the team is and how familiar the technology is to the teams.

Planning Phase

During this phase the team will set the priority order for the stories and an agreement of the contents of the first small release is made. The team first estimate how much effort each story requires and the schedule is then agreed upon. Notice that the time span of the schedule of the first release should not be more than exceed two months and the planning phase itself takes a couple of days.

Iterations to release Phase

In this phase the schedule set in the planning stage will be broken down to numbers o iterations that each takes one to four weeks to implement the code. The first iteration should create a system with the architecture of the whole system. This is achieved by selecting the stories that will enforce building the structure for the whole system. Also notice that the customer would decide what stories will be implement for each iteration. Lastly the functional tests create by the customer are run at end of the every iteration.

Productionizing phase

This phase requires extra testing and checking of the performance of the system before the system can be released to the customer. At the phase , new changes may still be found and decision would need to be made if they should be included in the current release. During this phase, the iterations may need to be quickened from 3 to 1 weeks. The ideas and suggestions will be documented for later implementation during, eg the maintenance phase.

Maintenance phase

This phase is to keep the system in the production running while also producing new iterations. In order to do this , it requires an effort also for customer support tasks. Thus, the development speed may slow down after the system is in production. This phase may also require incorporating new people into the team and changing the team structure.

Death phase

This phase is when the customer nearly have no longer any other stories to be implemented. This requires that the system satisfies customer needs also in other respects. Death may also means if the system is not delivering the desired outcomes, or if it becomes too expensive for further development.

### Extreme Programming Activities

Coding

This in the most important phase in Extreme Programming life cycle. XP programming gives priority to actual coding over all other such as documentation to ensure that the customer receives something substantial in value at the end of the day.

Standards related to coding include:

Developing the code based on agreed metaphors and standards , and adopting a policy of collective code ownerships. Pair programming which means developing code by two programmers working together on a single machine, aimed at producing higher quality code at the same or lest cost. Strict adherence to 40-hour workweeks with no overtime. This ensures the developers work in the peak of their mental and physical faculties. Frequent integration of the code to the dedicated repository, with only one pair intergraing at a time to prevent conflicts, and optimization at the end.

Testing

This actives integrates testing with the development phase rather than at the end of the development phase. All codes have unit tests to eliminate bugs, and the code passes all such unit tests before release. Another key test is customer acceptance test, based on the customer specifications. Acceptance test run at the completion of the coding , and the developers provide the customer with the result of the acceptance tests along with demonstrations.

Listening

The basis of extreme programming is a continuous mechanism of customer involvement through feedback during the development phase. Apart from the customer the development also recives feedback from the project manager. The basis of feedback is the customer acceptance tests. Each feedback of the customer that specifies revised requirement becomes the basis of a new design, and the process of the customer that specifies revised requirement becomes the basis of a new design, and the process of design-coding-tests-listening repeats itself. If the customer remains satisfied with the test result the integration ends there, and the design for the new iteration starts, which again follows the design-coding-testing-listening cycle.

Designing

An iteration of XP programming starts with designing. The guiding principles of this stage are: Thrust on simplicity by expressing a thing only once and not adding functionality in anticipation. Using systems metaphor or standards on names, class names and methods, and agreeing on uniform styles and formats to ensure compatibility among the work of different team members. Using Software Class Responsibilities and Collaboration (CRC) Cards that allow for a departure from the traditional procedural mindset make possible object oriented technology. Such cards allow all members of the project team to contribute ideas, and collate the best ideas into the design. Create spike solutions or simple programs that explore potential solutions for a specific problem, ignoring all other concerns, to mitigate risk.

### Programming Values

Communication

* A poor communication in software teams is one of the root causes of failure of a project.
* Stress on good communication between all stakeholders—customers , team members and project managers
* Customer representative always on site
* Paired programming

Simplicity

* Do the simplest things that could possibly work
  + Implement a new capability in the simplest possible way
  + Refactor the system to be the simplest possible code with the current feature set.
* You aren’t going to need it
  + Never implement a feature you don’t need now.

Feedback

* Always a running system that delivers information about itself in a reliable way
* The system and the code provides feedback on the state of development
* Catalyst for change and an indicator of progress

Courage

* Project are people-centric
* Ingenuity of people and not any process that causes a project to succeed.

## The Strengths of Extreme Programming

### Test Driven Development

### Customer Integration

### Refactoring

### Pair Programming

The Weaknesses of Extreme Programming

### Customer Integration

### Undocumented Requirements

### Organisational Size

### Organisational Readiness

### Minimal Management Principles