

**Imus Institute
of Science and Technology**

UNDERGRADUATE THESIS

Assessment Software

Author:

John Jeruel TIONGCO

Adviser:

Engr. Berone BUNYI



*A thesis submitted in fulfillment of the requirements
for the degree of BS in Electronics Engineering*

in the

Engineering Department

August 3, 2021

Declaration of Authorship

I, John Jeruel TIONGCO, declare that this thesis titled, "Assessment Software" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

**Imus Institute
of Science and Technology**

Abstract

Engineering Faculty
Engineering Department

BS in Electronics Engineering

Assessment Software

by John Jeruel TIONGCO

The project aims to explore the possibility of modernizing assessment procedures via software developed using industry standards such as AGILE, UP, UML and many others. The paper can treated as a *blue print* and the *alpha version* of the software a *proof of concept*. The paper also contains many software development artifacts such as a *use case model*, *class diagrams*, *interaction diagrams*, and many other models.

This project does not aim to, first, justify the existence of the project by presenting a robust *business case*. Second, produce a software up to its *end of life-cycle*.

Acknowledgements

First, I would like to thank Engr. Berone Bunyi, my advisor, for guiding the project and providing support at every turn.

To Engr. Arnold Balbuena as my mentor and father of the engineering department.
To Engr. Gilbert Laguardia for providing feedback during initial presentation.

To Aleli Benzon for proof reading and providing suggestions to improve readability.

Contents

Title	i
Declaration of Authorship	iii
Abstract	v
Acknowledgements	vii
Contents	ix
List of Figures	xi
List of Tables	xiii
List of Abbreviations	xv
Preface	xix
1 Inception	1
1.1 Vision	1
1.1.1 Introduction	1
1.1.1.1 Reference	1
1.1.2 Positioning	1
1.1.2.1 Problem Statement	1
1.1.2.2 Product Position Statement	2
1.1.3 Stakeholders and User Descriptions	2
1.1.3.1 Stake Holder Summary	2
1.1.3.2 User Summary	2
1.1.3.3 User Environment	2
1.1.3.4 Key Stakeholder or User Needs	3
1.1.3.5 Alternatives and Competition	3
1.1.4 Product Overview	4
1.1.4.1 Product Perspective	4
1.1.4.2 Assumption and Dependencies	4
1.1.5 Product Features	4
1.1.6 Other Product Requirements	4
1.2 Business Case	5
1.2.1 Rationale for Leaving Out Business Case	5
1.3 Use Case Model	6
1.4 Supplementary Requirements	7
1.4.1 Introduction	7
1.4.1.1 Purpose	7
1.4.2 Functionality	7
1.4.3 Usability	7

1.4.4	Reliability	7
1.4.5	Performance	7
1.4.6	Supportability	7
1.4.7	Design Constraints	7
1.4.8	Online User Documentation and Help System Requirements	7
1.4.9	Purchased Components	7
1.4.10	Interface	8
1.4.11	Licensing Requirements	8
1.4.12	Legal Copyright and Other Notices	8
1.4.13	Applicable Standards	8
1.5	Iteration Plan	9
2	Elaboration Pt. 1	11
2.1	Use Cases Refined	11
2.2	Domain Model	11
3	Elaboration Pt.2 and Construction	15
3.1	Main Section 1	15
3.1.1	Subsection 1	15
3.1.2	Subsection 2	15
3.2	Main Section 2	15
A	Getting Started	17
B	Glossary	19
	Bibliography	21

List of Figures

2.1	Fully dressing UC1 to serve as reference for the domain model and design model.	12
2.2	Fully dressing UC1 to serve as reference for the domain model and design model.	13
2.3	Fully dressing UC2.	14

List of Tables

1	Resources Used in the Project	xix
---	---	-----

List of Abbreviations

LAH List Abbreviations Here
WSF What (it) Stands For

To Aleli, Bambam, Pimpim, and my Family, Thank you...

Preface

The Preface is an optional read but contains vital information on how the paper was structured, the resources used in creating the software and writing the paper itself, and finally additional resources such as the online repository of the project.

Structure of the Paper

Although the paper is labeled as thesis it is virtually a capstone project. This is more evident when the only requirement given is that it should have a prototype. And so the suggested format from the article “Guidelines in Writing Design Project / Research Project Proposal” doesn’t seem to frame the project properly. Be that as it may, in order to comply and frame the project properly, I superimposed the *Universal Process Phases: Inception, Elaboration, Construction, and Transition* over the suggested format in the following manner:

Chapter I–The Problem and Its Setting *superimposed with Inception* – Contains the subsections *Vision, Business Case, Supplementary Requirements, Use-case Model, and Iteration Plan* which replaced the subsections *Introduction, Statement of the Problem, Objectives of the Study, Importance of the Study, and Scope and Limitations of the Study* respectively. The lack of the subsections *Definition of Terms* and *Conceptual Framework* is addressed on the next chapter.

Chapter II–The Problem and Its Setting *superimposed with Elaboration Pt. 1* – This chapter completely deviates from the suggested format since there are no hypothesis to prove or a conclusion to draw. There are also no conceptual framework on which the project can be framed into. Instead this chapter is spent on creating that *conceptual domain* with the help of a *class diagram* in **UML**.

Chapter III–Methodology *superimposed with Elaboration Pt.2 and Construction* – description

In addition the manual of the software can be found in “Cross Ref” appendix

Software, Standards, and Technology Used

Below is a table enumerating all the software, standards or languages, and other technologies used in the creating the software and the paper itself.

TABLE 1: Resources Used in the Project

Resource	Function	Link
Qt5	UI IDE	some link

Repository

Repo Link

Chapter 1

Inception

Objectives

- Establish the vision, and scope of the software
- Provide the artifacts created during the inception phase such as the Vision, Use Case Model, Business Case, etc...
- Justify why some artifacts have been dropped off
- Create a foundation for the baseline architecture

1.1 Vision

Revision History

Date	Version	Description	Author
June 20, 2021	Draft	Initial vision for the project.	John Jeruel TIONGCO

1.1.1 Introduction

The purpose of this document is to collect, analyze, and define high-level needs and features of the *Assessment Software*. It focuses on the capabilities needed by the stakeholders, and the target users, and why these needs exist. The details of how the *Assessment Software* fulfills these needs are detailed in the use-case and supplementary specifications.

1.1.1.1 Reference

See Glossary [B](#)

1.1.2 Positioning

1.1.2.1 Problem Statement

problem of	The lack of modernization in assessment
affects	Test makers and test takers
the impact of which is	That there is too much manual labor that can otherwise be automated through software.
a successful solution would be	Create a software that can automate assessment, reducing time spent and improving the integrity of the results.

1.1.2.2 Product Position Statement

For	Individuals or small to medium size organizations.
Who	Wish to automate assessment procedures.
Assessment Software	Is an <i>open-source</i> software product.
That	That can automate assessment such as testing and evaluating.
Unlike	Expensive web-base XaaS solutions which are complicated to use. Or general document editing programs which lacks essential features.
Our Product	Is simple. Doesn't require programming skills to use or extensive hardware resources. Can be used <i>off-line</i> . And intended for personal and small scale uses.

1.1.3 Stakeholders and User Descriptions

1.1.3.1 Stake Holder Summary

Name	Represents	Role
Developer	As of now represents the entirety of the project	Main source of funds and does everything in the project, from requirement, analysis and design, implementation, testing, deployment to project management
End-user	Represents individuals who wants to use the software, mainly me	Uses the software

1.1.3.2 User Summary

Name	Description	Stakeholders
Test makers	Creates question banks and generates test Administers test.	Self-represented
Test takers	Takes the test	Self-represented
Evaluators	Evaluates the results and draws conclusions.	Self-represented

1.1.3.3 User Environment

Assessment is a lengthy process and tedious at times. Often the purpose of assessment is missed altogether and *passing the test* becomes the priority. This leads to assessment being reduced to a compulsory *formality* instead of a tool for learning.

Users are expected to have at least a relatively *low-end PC* in order to use the software. Internet connection is not required in the operations but might be needed in updating the software.

1.1.3.4 Key Stakeholder or User Needs

The *needs* provided in these section is generated by myself as I am the first *end-user* of the software. As it moves further in development I might elicit new needs from future stake-holders and end-users.

Need	Priority	Concerns	Current Solution	Proposed Solution
Automatic test creation	High	Manual test creation is tedious and time consuming	No current solution	Employ the use a question bank and automatically generate a test questionnaire
Automatic test checking	High	Software should be able check test answers	No current solution	Answers are of course included with test item so checking the answer can be automated
Digitized test results	Low	Manual recording is time consuming and laborious	No current solution	Test results are converted to a spread sheet readable format like .csv or can be pipe-lined to a DBMS . alternatively an evaluation module can made for the software
User Management	Medium	User should be able to save their details along with their work	No current solution	Simple user class that can be used to save data about the user
Able to run on a low-end PC	Medium	I have a potato laptop	No current solution	Develop for low-end PC
Usability Requirements	Low	—	No current solution	Will be addressed in the <i>UI Mock-ups</i>
Technical support needs	Low	—	No current solution	Will be addressed during deployment stage

1.1.3.5 Alternatives and Competition

The only true open-source program that can be considered a competition for this project is a *computer-based testing* software called **TCEExam**. TCEExam is a platform

type software that must be run using a web-server. I wouldn't deny the fact that TCExam outclasses this project but in my opinion it fulfills a different niche than what Assessment Software is trying to fill.

For alternatives, there are open-source documents editors such as **Libre Office: Writer, Google Docs and Google Forms**, etc These software are intended for general use and lack the main features that address the needs of the end users.

1.1.4 Product Overview

1.1.4.1 Product Perspective

Assessment Software is planned to be a standalone software although it would use Qt5 GUI API for its *UI* it's back end architecture would be designed from scratch.

1.1.4.2 Assumption and Dependencies

No dependencies to other systems as of this version.

1.1.5 Product Features

Will be provided at Chapter [2](#)

1.1.6 Other Product Requirements

None for this version

1.2 Business Case

One of the goals in the inception phase is to determine whether a new system is feasible and worth exploring. “feasible” is straight forward, while “worth exploring” is somewhat vague. To clarify the term, by saying a project is worth exploring is tantamount to saying the project is profitable. By creating a *business case* both those **risk**, as they are termed in the industry, are analyzed and ultimately justified. But as it stands I find that building a *business case* for this project to be redundant.

1.2.1 Rationale for Leaving Out Business Case

The rationale for leaving out *business case* is that both questions asked in this section already has answers. Yes, the project is feasible, it has to be, otherwise the project would be canceled¹. Is the project profitable? The question is invalid since the project is meant to be *open-source*. Now the question is “Is it valid to leave out the building the business case for the project?”. Yes, it is perfectly valid². An artifact’s function is to serve as utility for the project or the development of a program. In this case building a *business case* doesn’t provide any utility and will only consume resources.

In addition *business cases* are a project by themselves. It may contain sections such as *executive summary*, *problem statement*, *analysis of the situation*, *solution options*, *cost-benefit analysis*, etc.... It is best that such studies and analysis are left to those who have expertise in the subject matter³. As an engineering student I think such endeavors is out of my field of study.

¹The paper is an undergrad course requirement. The student of course has to fund it to complete the course.

²biblio craid larman

³Say students of business disciplines such as BS Accountancy or BS Business Management

1.3 Use Case Model

Revision History

Date	Version	Description	Author
June 20, 2021	Draft	Casual form for use cases.	John Jeruel TIONGCO

UC1: Create Test

Main Success Scenario: User starts the system. Assume that the user has already entered his details in the system. Assume also that a question bank has been provided by user. The system generates a test as per specification of the user e.g. number of sections, items per sections, type of test for each sections, etc The user saves the generated test.

Alternate Scenario: The user has not entered his details. The user then goes to *subfunction* Create New User or *subfunction* Change User. No question bank to generate from. The user goes to the *subfunction* Create a Question Bank. Cannot generate test because question bank does not meet the specification the user has given the system. System provides hints in order to proceed with the generation.

UC2: Take Test

Main Success Scenario: User starts the system. Assume that the user has already entered his detail into the system. Assume that a generated test file is available. The system loads up the test file. The user takes the test on the systems UI or other means of output. The test result is saved along with the user details.

Alternate Scenario: The user has not entered his details. The user then goes to *subfunction* Create New User or *subfunction* Change User. The user wants to check her work. *Subfunction* Check Test is called.

UC3: View Scores

Main Success Scenario: User starts the system. Assume that the user has already entered his detail in the system. The system retrieves the data requested by the user e.g. test results with the user details of test taker. The systems views the scores in a comprehensive manner e.g. graphs or reports.

Alternate Scenario: User wants to collect the data instead of viewing it on the UI of the system or UI for viewing test result is not yet developed since it is lower in priority. *Subfunction* Other Results Output is called.

UC4: Manage Users

Main Success Scenario: Is a collection of subfunction-level cases such as Create New User, Delete User, and Edit User Details.

Alternate Scenario: When subfunction other than Create User is called.

1.4 Supplementary Requirements

Revision History			
Date	Version	Description	Author
June 20, 2021	Draft	Supplementary requirements during inception phase.	John Jeruel TIONGCO

1.4.1 Introduction

1.4.1.1 Purpose

The purpose of this document is to define requirements of the Assessment Software. This Supplementary Specification lists the requirements that are not readily captured in the use cases of the use-case model. The Supplementary Specifications and the use-case model together capture a complete set of requirements on the system.

1.4.2 Functionality

Functional requirements are captured via the defined use cases.

1.4.3 Usability

Ease of Use – Using the software should not require any knowledge in programming. UI should be intuitive and akin to many popular software.

1.4.4 Reliability

Redundancy – Test results and question banks should have redundancy in case of system failure to prevent loss of critical data.

1.4.5 Performance

Low Resource Requirement – Software should not consume too much resources. Functionality is prioritized over aesthetics.

1.4.6 Supportability

Availability of Updates – Updates should be readily available via repository.

1.4.7 Design Constraints

Software must be able to run on low end PCs

1.4.8 Online User Documentation and Help System Requirements

Getting Started Document – At least a “Getting Started” documents should be provided.

Man Pages – Man pages style Help for functions while UI is not yet developed.

1.4.9 Purchased Components

None as of this version.

1.4.10 Interface

No UI plans as of the first iteration.

1.4.11 Licensing Requirements

Should be open source. Publicly license may be used.

1.4.12 Legal Copyright and Other Notices

None as of this version.

1.4.13 Applicable Standards

None as of this version.

1.5 Iteration Plan

For now I am the sole developer of the software. I find it absurd to create a schedule for each task. With regards to the timeline it also difficult to provide one with any degree of accuracy. As for milestones below is list of critical ones going into elaboration phase of *iteration 1*.

- Create domain model for the core architecture
- Convert UC1 and UC2 to fully dressed version. To define requirements even further.
- Implement basic key scenario mainly UC1.
- Implement *Start Up* case for initialization needs of the system.
-

Chapter 2

Elaboration Pt. 1

Objectives

- Establish domain class model
 - Code first design model
-

2.1 Use Cases Refined

In order work *Iteration 1: Elaboration Phase* the requirements should be first refined and this can be achieved by fully dressing *UC1 and UC2*, the main use-cases that will be the subject of the Design Model and Implementation.

Keep in mind that these refined use cases are not separate from the use case model artifact. In fact they are updates to the prior presented use case model and is reflected in the doc folder of the repository. It's isolation from the model only serves to improve the narrative of the project.

Refer to figure 2.1 for UC1 and figure

2.2 Domain Model

Domain model package contains the Domain Class Model and its Domain Class Diagram. the main purpose of this model to serve as reference for the design class model which in turn will be map to the source code of the system.

UC1: Create Test

Scope: Assessment Software

Level: User Goal

Primary Actor: End-User

Stake Holders and Interest:

–User: Wants less manual labor. Wants organized and custom layout. Wants intelligent options in multiple-choice types. Wants no redundancy in stems.

–Developer: Wants to mitigate complexity in design.

Preconditions: User has already “logged in”. Question bank is available.

Success Guarantee (or Postconditions): Test generated without error.
Generated test saved. Generated test can be loaded.

Main Success Scenario (or Basic Flow)

1. User enters a title for the test.
2. User chooses a question bank file to generate from.
3. User enters number of sections for the test.
4. User configures number of stem for the first section.
5. User configures type of stem for the first section.
6. Repeat step 4–5 until all sections are configured.
7. User starts automatic generation.
8. No errors detected. Test generated is viewed.
9. Generated test is saved.

FIGURE 2.1: Fully dressing UC1 to serve as reference for the domain model and design model.

Extensions (or Alternative Flow)

a*. At anytime the system fails.

To support recovery, ensure that question bank is safe from corruption.

1. Error report is logged.
2. User restarts the system.

3. User is informed of the system failure with the error message.

2a. No question bank available. User enters subfunction-level *Create Question Bank* use case.

2b. Question bank has insufficient stem to populate section. System throws exception “not enough stem to populate.”

4a. User wants to customize subsection. Aside from the number of items and type of question the user can also chose the topic via “tags” and the level.

8a. Errors is detected.

1. Not enough question to fill the section. System suggest adding more stems or using multiple question banks.

2. Not enough option to fill stem options. System suggest use of “dictionary”.

Special Requirements:

None for this version.

Technology and Data Variation List:

None of this version.

Frequency of Occurrence

Often. Considered main goal of actor.

Open Issues:

– Will be addressed during initial construction.

FIGURE 2.2: Fully dressing UC1 to serve as reference for the domain model and design model.

UC2: Take Test

Scope: Assessment Software

Level: User Goal

Primary Actor: End-User

Stake Holders and Interest:

–User: Wants two types of test mode. *Mode 1*: One question then followed by showing the answer. *Mode 2*: Answer all question then show all answers or no answers shown. Wants good UI (will address needs during UI design.)

Preconditions: User has already “logged in”. Generated test is available.

Success Guarantee (or Postconditions): All stem response is saved. Results of test saved.

(A) Fully dressing UC2.

Main Success Scenario (or Basic Flow)

1. User loads generated test.
2. User chose mode 1.
3. System starts with first section and views the first stem.
4. User enters response for stem. System saves response.
5. System shows answer. Systems compares answer and response. If match then system informs user that response is correct. Else system informs user that response is wrong.
6. System goes to next stem. Repeat step 4 – 5.
7. Repeat step 6 until all stems for all sections have been responded.
8. System computes results.
9. System saves results.

Extensions (or Alternative Flow)

a*. At anytime the system fails.

To support recovery, ensure that test results are safe from corruption.

1. Error report is logged.
2. User restarts the system.
3. User is informed of the system failure with the error message.

1a. No question bank available. User enters use case *Create Test*.

2a. User chooses mode 2.

1. System displays all stems in first section.
2. User enters response for all stems in section.
3. System views next section and all its stem.
4. Repeat step 2–3 until all section is answered.
5. If view answers is available user may view answers. Else continue.
6. Go to step 8–9 of *Main Success Scenario*

4a. User does not know answer.

1. User may skip the stem. 2. Before moving to next section system informs user to review unanswered stem.
3. If user chooses to review stem the system re-displays stems without response. Else go to step 6 of main scenario.

Special Requirements:

None for this version.

Technology and Data Variation List:

None of this version.

Frequency of Occurrence

Often. Considered secondary goal of actor.

Chapter 3

Elaboration Pt.2 and Construction

3.1 System Sequence Diagram

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

3.1.1 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

3.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

3.2 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Appendix A

Getting Started

Write your Appendix content here.

Appendix B

Glossary

Write your Appendix content here.

Bibliography

- Arnold, A. S. et al. (Mar. 1998). "A Simple Extended-Cavity Diode Laser". In: *Review of Scientific Instruments* 69.3, pp. 1236–1239. URL: <http://link.aip.org/link/?RSI/69/1236/1>.
- Hawthorn, C. J., K. P. Weber, and R. E. Scholten (Dec. 2001). "Littrow Configuration Tunable External Cavity Diode Laser with Fixed Direction Output Beam". In: *Review of Scientific Instruments* 72.12, pp. 4477–4479. URL: <http://link.aip.org/link/?RSI/72/4477/1>.
- Wieman, Carl E. and Leo Hollberg (Jan. 1991). "Using Diode Lasers for Atomic Physics". In: *Review of Scientific Instruments* 62.1, pp. 1–20. URL: <http://link.aip.org/link/?RSI/62/1/1>.