Early Earthquake and Tsunami Waning Viewer

Yicheng Shao (Eason)

September 7, 2024

Abstract

Give a brief summary outline of your project.

Contents

1 Analysis

1.1 Problem Area

Define the general problem area that your project covers.

1.2 Client and End User

A client is the person who has commissioned the system. They may be the same person as the end user or there may be additional end users. If there are multiple end users, do they all have the same needs/requirements?

A **real** client/user is beneficial to keep things realistic and to mirror a real software development project.

Some background as to the client/user will be required.

- Who are they?
- What is their background?
- What is their level of experience in the problem area being undertaken?

  Novices/experts will have different requirements.

If you don't have a specific user you should still have an intended **target audience/user base** in mind.

1.3 Research Methodology

Describe **how** you went about investigating the requirements. This may include a range of measures:

- Investigation of similar systems
- Web research for key concepts/algorithms
- Client/end user interview
- Questionnaires to potential end-users of the system

1.4 Features of proposed solution

As a result of research, you should identify the key features (in general terms) that your system will have:

- List of key features that will be Required
- Discussion of the scope and potential limitations to the system given the time constraints.

1.5 Requirements Specification

The requirements specification is a document/contract with the client that outlines what you will deliver. The contents need to have SMART (specific, measurable, achievable, realistic, timely) goals.

After your system has been completed you will need to test against this.

Requirement №	Description	Success Criteria	Measurement Method

Table 1: Table of Requirements.

1.6 Critical Path

Order of development for the tasks that will need to be completed. This may reflect an iterative approach to software development. Software development will be undertaken using an *agile* methodology as opposed to a *waterfall* model of software development. It is expected that development will go through a number of iterations that will add increasing functionality to the system.

2 Design

Algorithms + Data Structures = Programs.

2.1 Hierarchy Chart

A top-down approach to problem solving will lead to the identification of tasks with sub-tasks. i.e. modules and functions required. This shows how **decomposition** is applied.

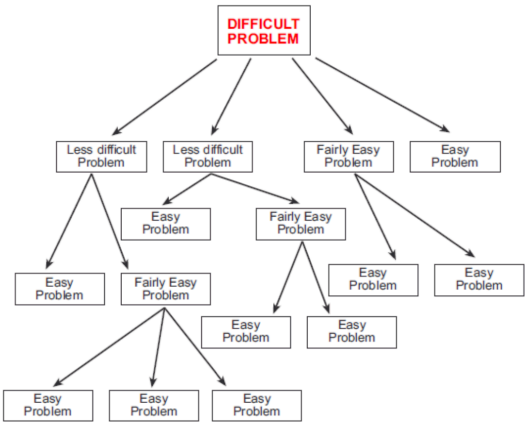

Figure 1: Hierarchy Chart.

2.2 Data Structures/Data modelling

2.2.1 External Data Sources

If you are scraping/gathering data from APIs from external sources you should define the relevant format/parameters.

2.2.2 OOP Model

OOP modelling (classes, methods, attributes, inheritance etc.). Class diagrams would be useful (these are covered in Bond book 1 page 185 onwards). Diagrams should follow conventions for inheritance/composition and private/protected/public methods/attributes.

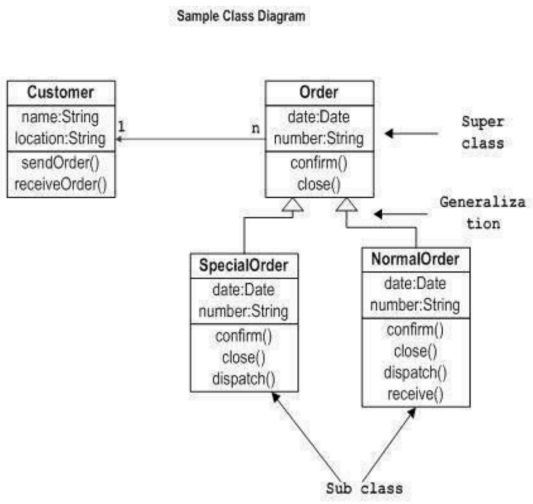

Figure 2: Class Diagram.

2.3 User Interface

You will need to draw up a prototype for the user interface. You may do this within the software package you implement your solution in.

- Screen designs
- Menu options/sequences
- Buttons/keys/commands (command line)

2.4 Hardware Software Requirements

Draw up a hardware and software specification for items that are required.

3 Technical Implementation

3.1 Key Code Segments

3.1.1 Data structures

Implementation of ADTs and OOP Classes to be demonstrated.

3.1.2 Modularity

Code should be created and tested in separate modules that are integrated later. Use sub-headings for each module, define the purpose of the module, and show unit testing of the module.

3.1.3 Defensive Programming/Robustness

Exception handling

4 Testing

Consider how you will test your project. You should devise a test strategy that encompasses a range of methods.

4.1 Test Strategy

- Unit testing (of individual functions)
- Integration testing (e.g. different modules/class files)
- Robustness (demonstrating defensive programming skills/exception handling)
- Requirements testing (against your initial requirements - a table with test number, description, test data, expected result, evidence (screenshot/video time link) would be suitable)
- Independent end user beta testing (this will assist with your evaluation)

4.2 Testing Video

- You can include a video to assist (but you will need to reference the timepoint at which relevant evidence appears)
- If you include a video you will need to have it publicly available.
- It is suggested that you include a QR code in your testing to give a link to it the video (for the moderator) rather than just giving a long URL on its own.

4.3 System Tests (against original requirements specification)

You need to give evidence in support of requirements that have been met e.g. reference to a relevant test/screenshot/relevant code.

Requirement №	Description	Success Criteria	Tests + Evidence

Table 2: Table of Tests.

5 Evaluation

5.1 Requirements Specification Evaluation

Personal evaluation

- Copy and paste your original requirements from your project analysis
- You need to review each requirement and comment objectively on whether it was *fully met/partially met/not met*.

Requirement №	Description	Success Criteria	Fully/Partial/Not met (Reflective Comment)

Table 3: Table of Evaluation.

5.2 Independent End-User Feedback

End user/client evaluation

- there **must** be meaningful end user feedback
- You should hold a review meeting with your end user
- Write down any key feedback that they give you. E.g. Agreement that a particular requirement has been meet/comments as to aspects that they find sub-optimal/comments as to additions they would like to see

Requirement №	Description	Acceptance Y/N	Additional Comments

Table 4: Table of Feedback.

5.3 Improvements

You need to give consideration to a number of potential future improvements that could be made. They may arise from either your experience or from feedback given to you by your end user. Ideally at least one should be in response to end user feedback.

- Write a paragraph for each potential improvement/change
- The improvements/changes could result from additional functionality that has been identified as being beneficial or could be as a result of required efficiencies if some processes are clunky or require faster run-times
- You should then comment on how the proposed change could be implemented moving forward. i.e. what would need to be changed/developed and how? You are not expected to actually make any changes; just comment on the possibilities.

6 Code Listing