Project Management
Project Report

Chris O'Brien
0144266
Shane Whelan
09005763
Brian O'Donoghue

April 23, 2014

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Project Integration

2.1 Process Groups

2.2 Project Integration

2.2.1 Develop Project Charter

2.2.2 Project Management Plan

2.2.3 Directing and Managing

2.2.4 Monitoring and Controlling

2.2.5 Integrated Change Control

2.2.6 Close

Chapter 3

Project Scope Management

3.1 Project Scope

3.2 Requirements

3.3 Work Breakdown Structure

3.4 Validation and Control

Chapter 4

Project Time Management

4.1 Processes

4.2 Diagramming

4.3 Task Dependencies

Chapter 5

Project Cost Management

5.1 Budgeting

5.2 Realistic Targets

5.3 Cost Control

Chapter 6

Project Quality Management

The aim of Quality Management is to manage the quality of the final software product, and the quality of the development process used to create it. Another measure of quality is ensuring that the final product meets the requirements as defined by the stakeholders.

6.1 Quality Process

The process following for quality would focus on three areas.

1. Plan Quality Management
2. Perform Quality Assurance
3. Control Quality

6.1.1 Quality Management

Quality Management is the process of ”identifying quality requirements and standards for the project” (PMBoK 2000). It is also important that the project demonstrates ”compliance with relevant quality requirements and standards” (PMBoK 2000). This project focuses on a number of quality attributes during the development process.

1. Performance
    - The ability of the project to respond to requests with clearly defined time constraints. For this project, the reaction time the the web application to user demanded is important.
2. Security
    - The protection, integrity and non-repudiation of stored data. User information, such as payment information, will be stored. The security of this data is integral to the application.
3. Usabilty
    - The system must be easy to use, and easy to learn. Usability studies need to be performed on the application to ensure ease of use.
4. Maintainability
    - It must not be cost prohibitive to perform corrective, perfective and adaptive maintenance on the product. The project must be designed in such a way that any changes that need to be made can be done at minimal cost.
5. Re-usability
    - Components of the system must be design modularly in order to promote reuse. A reusable system, if the risk is managed correctly, will save costs if components are reusable, and may also allow further product lines.
6. Portability
    - The project application must work across a range of operating systems and browsers.

The project aims to conform to the ISO 25010 standard detailed later in this report.

6.1.2 Quality Assurance

Quality Assurance is governed by the user of standards, such as the ISO 25010, within the project. In a big organisation, the idea of Total Quality Management is used to ensure that a high quality product is delivered. This draws on a number of different quality methods illustrated in Figure 6.1.

- Controls

- Job Management
- Well defined processes
- Employee competence
- Organisation culture
- Team spirit

Figure 6.1: Sample of methods to ensure quality

6.2 Quality Control

Quality Control is a monitoring and control process. It inspects every project deliverable, and is measured in some way. Results are checked to ensure their conformity to quality standards. This process covers the project, and the products produced by the project. Corrective measures needs to be applied to any defects discovered throughout this process.

Quality Assurance can often be confused with Quality Control. The PMBoK gives an example to clarify the difference.

"Quality assurance could be calibrating a machine or training the operator, while quality control is inspecting or testing the products that are being made by the machine" (PMBoK 2000).

6.3 Quality Support

Quality Support throughout the development process was given through the use of support tools, namely inFusion, that examined code focusing on a number of quantifiable attributes. These included Encapsulation, Coupling, Cohesion, Complexity and Inheritance. This allowed developers to be aware of design flaws throughout the development process, and allow the identification of bugs and discrepancies at development time, rather than expose them during testing.

6.3.1 Tools and Techniques

Techniques

A number of defined techniques are available to use within the project in order to control quality.

Cost Benefit Analysis offsets the costs of supporting a quality attribute versus the benefit of implementing it. An example may be having a very strong security component, when little to no data is stored in the application.

Cost of Quality "includes all costs incurred over the life of the product by investment in preventing non-conformance to requirements" (PMBoK 2000). The types of failures are internal and external: those found by the project, and those found the the customer. These need to be factored into overall project cost, and decisions made regarding quality can greatly impact the project long term.

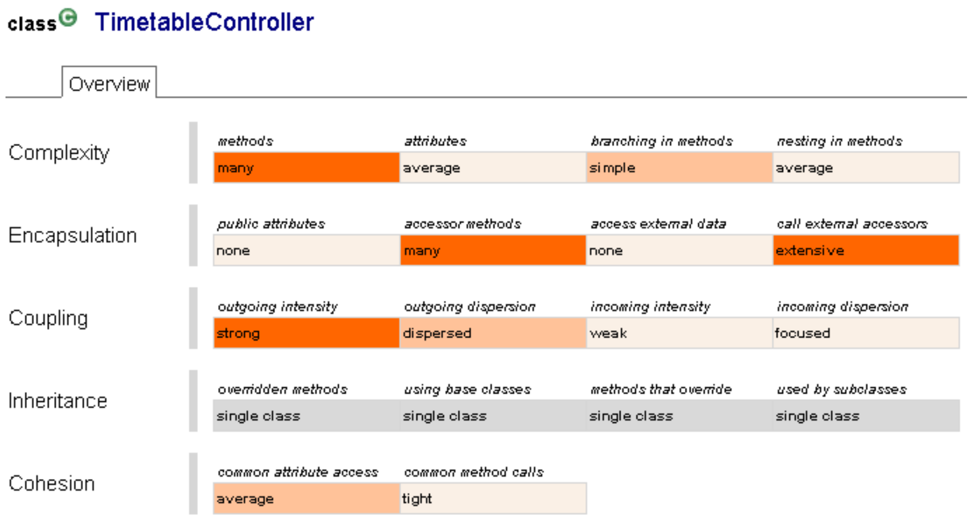
A number of diagrams can be used to map quality within a project, shown in Figure 6.2.

- Cause-and-effect Diagrams
- Flowcharts
- Check sheets
- Pareto Diagrams
- Histograms
- Control Charts
- Scatter Diagrams

Figure 6.2: Quality Diagramming Options

Tool - Infusion

Infusion is a tool that highlights flaws in a number of areas. It breaks them now, and identifies 'bad code smells' (Fowler 2006). Figure 6.3 shows a breakdown of a prototype application, and shows possible quality deficits within the application. This is an overall breakdown on the entire system. This can be further broken down to class level.

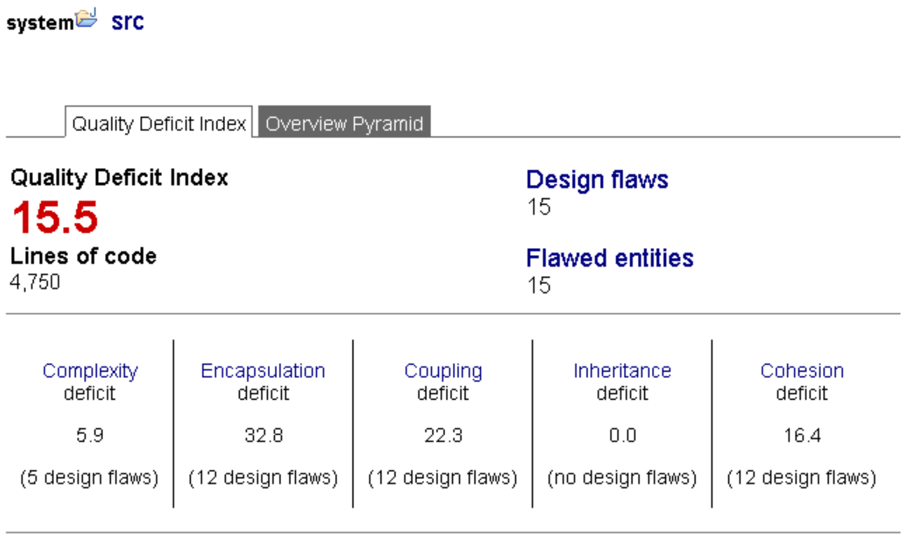

class TimetableController

Overview

Complexity	*methods*: many	*attributes*: average	*branching in methods*: simple	*nesting in methods*: average
Encapsulation	*public attributes*: none	*accessor methods*: many	*access external data*: none	*call external accessors*: extensive
Coupling	*outgoing intensity*: strong	*outgoing dispersion*: dispersed	*incoming intensity*: weak	*incoming dispersion*: focused
Inheritance	*overridden methods*: single class	*using base classes*: single class	*methods that override*: single class	*used by subclasses*: single class
Cohesion	*common attribute access*: average	*common method calls*: tight		

Figure 6.3: Example of Class 'Bad Code Smell' Breakdown using Infusion

An overall quality result is given to the application. Changes can then me made, in order to achieve a benchmark score. It is also worth noting that the use of frameworks, if that decision is made, made have a negative effect on these scores and a cognisance of this is paramount to ensure that time is not spend on corrective action for 'flaws' introduced by design choices.


system src

Quality Deficit Index | Overview Pyramid

Quality Deficit Index
15.5
Lines of code
4,750

Design flaws
15

Flawed entities
15

Complexity deficit	Encapsulation deficit	Coupling deficit	Inheritance deficit	Cohesion deficit
5.9	32.8	22.3	0.0	16.4
(5 design flaws)	(12 design flaws)	(12 design flaws)	(no design flaws)	(12 design flaws)

Figure 6.4: Quality Measurement using Infusion

6.4 Quality Standards

One standard looked at in this project was *ISO/IEC 25010*. This replaced the old model, *ISO 9126*.

In this model, there is a focus on a number of different attributes. In order to help a project succeed, there attributes must be examined, and acted upon, through the project management process. This is to ensure that the final product is a high quality product, and one that is what the customer wanted. Figure 6.5 shows a summary of the areas that this standard focuses on.

1. Functional Stability
    - This refers to the "appropriateness, accuracy and compliance" of the final product. (ISO 2011)
2. Reliability
    - This section focuses on the "availability, fault tolerance and recoverability compliance" of the final product. (ISO 2011)
3. Performance Efficiency
    - This is concerned with measurements such as "time behaviour and resource utilisation compliance". (ISO 2011)
4. Operability
    - This encompasses usability. and focuses on "ease of use, helpfulness, attractiveness, technical accessibility and learn-ability" of the created system. (ISO 2011)
5. Security
    - Within this application, the focus of security is defined by the user requirements. This area will focus on the "confidentiality, integrity, non repudiation, and authenticity" of information within the system.
6. Compatibility
    - This area focuses on the effect that different configurations may have on the system. This include "interoperability and replace-ability". (ISO 2011)
7. Maintainability
    - This attribute focuses on the design of an application. Key attributes are the "modularity, re-usability, testability and changeability" of the application (ISO 2011). These attributes normally increase the development time of the application, and thus the risk and cost.
8. Transferability

- This refers to the ”portability and adaptability” of the application to run on different systems, operating systems and configurations. (ISO 2011)

Figure 6.5: Quality Areas Defined by ISO 25010

Chapter 7

Project Risk Management

Project Risk Management is defined by PMBoK as ”the process of conducting risk management planning, identification, analysis, response planning and monitoring and control on a project” (PMBoK 2000). The objectives of this project management area are to increase the probability and impacts of positive events, while at the same time decreases the same of negative events within the scope of the project.

7.1 Processes

There are a number of processes involved in the Risk Management aspect of a project plan, illustrated in Figure 7.1.

1. Plan Risk Management
    - The process of ”defining how to conduct risk management activities for a project” (PMBoK 2000).
2. Identify Risks
    - Determining ”which risks may affect the project and documenting their characteristics” (PMBoK 2000).
3. Perform Qualitative Risk Analysis
    - This is the process of prioritising risks. These risks can then be sent ”for further analysis or action by assessing and combining their probability of occurrence and impact” (PMBoK 2000).
4. Perform Quantitative Risk Analysis

- This is the process of ”numerically analysing the effect of identified risks on the overall project objectives” (PMBoK 2000).

5. Plan Risk Responses
   - This builds on the results of the previous processes. This process plans for how each risk should be managed, and which person is responsible for the management of that particular risk.
6. Control Risks
   - This is the process of ”implementing the risk response plans, tracking identified risks, identifying new risks, and evaluating risk process effectiveness throughout the project” (PMBoK 2000).

Figure 7.1: Risk Management Processes

One of the main risks in a software project is the choice of the architecture. A qualified software architect, with the experience necessary to properly evaluate and choose a software solution, would play a major role in negating this risk. The choice of a proper architecture lays the groundwork for the future development, and is a mistake that is very hard to rectify further on in the project lifecycle.

7.2 Techniques

7.3 Risk Management Plans

The main risks exposed by this project are design choices and requirements elicitation. A big risk in the project would be a choice to develop a more generic solution, and tailor it towards the client, rather than develop specifically for the client. This approach would cost more over the lifetime of the project, but it would also allow the project to reach a larger potential customer base. The design of the system as a more modular, separate unit, composed of many parts, would increase development time and complexity, but long term, it would allow for reduced costs with regards towards the various kinds of maintenance: corrective, adaptive and perfective.

1. Use of Commercial Off The Shelf (COTS) software within the application
   - Is there a cost to the user to get this software? What happens if it’s updated/discontinued?

Chapter 8

Project Human Resource Management

8.1 Roles

8.2 Responsibility

8.3 Authority

8.4 Identify Competencies

8.5 Organisation and Planning

References

Fowler, Martin (2006). *Code Smell*. URL: `http://martinfowler.com/bliki/CodeSmell.html`.

ISO, ISO (2011). “IEC 25010: 2011: Systems and software engineering–Systems and software Quality Requirements and Evaluation (SQuaRE)–System and software quality models”. In: *International Organization for Standardization*.

PMBoK, A (2000). “Guide to the project Management body of knowledge”. In: *Project Management Institute, Pennsylvania USA*.