[Note: This is a sample/template document for the Software/System Quality Assurance Plan (SQAP), from a number of years ago. Please adapt and adjust to your project needs and add details.]

[PROJECT TITLE]

System Quality Assurance Plan (SQAP)

[TEAM NAME]

List of your Names:

Name	Position	email	phone

SUBJECT CODE, NAME, SEMESTER AND DATE

Review history

Version	Date	Author	Comments
1.00		All Authors	Created initial draft document.
1.10		All Authors	Updated based on feedback from external reviews.
1.20		S. Smith	Updated based on internal review and Supervisor feedback.
1.30		S. Smith & T. Le	Updated based on internal review and Supervisor feedback.
1.40		All	Final Submission.

Acronyms/Abbreviations

ASAP As Soon as Possible

COB Close of Business (5:00 PM)

DMO Defence Materiel Organisation

DMS Data Management System

GUI Graphical User Interface

IEEE Institute of Electrical and Electronics Engineers

JDK Java Development Kit

JRE Java Runtime Environment

Ver. Version

SQAP Software Quality Assurance Plan **SRS**

Software Requirements Specification

SVN Subversion

SVVP Software Verification and Validation Plan

Contents

Chapter 1

Introduction

1.1 Author List/Roles

Author	Student ID	Role Semester 1	Role Semester 2
		Team Leader / Supervisor Liaison	Testing/Usability Champion
		SVN / Mantis Champion	Documentation / Quality Champion
		Client Liaison	N.A (No longer part of Team)
		Usability/Quality Champion	SVN / Mantis Champion
		Documentation Champion	Team Leader / Supervisor Liaison
		Coding Champion / Sargent At Arms	Coding Champion / Client Liaison

1.2 Purpose

This document outlines the policies and procedures that members of Team 02 will follow to achieve an overall high standard of quality for Project Eagle, a Data Management System (DMS) for the Defence Materiel Organisation (DMO). All team members are expected to adhere to the processes outlined in this document.

Chapter 2

Reference Documents

- Institute of Electrical and Electronics Engineers (IEEE) Std 730-1998, IEEE Standard for Software Quality Assurance Plans
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- SPINGRID Software Quality Assurance Plan, Version 0.1.3 14 June 2006
- OMP Quality Assurance Plan, Version 1.1 May 15, 2006
- Swinburne Java Coding Standard - SwinBrain
- Swinburne .NET Coding Standard - SwinBrain
- SVN Best Practices: http://svn.apache.org/repos/asf/subversion/trunk/doc/user/svn-best-practices. html

Chapter 3

Management

3.1 Organisation/Roles

The following list contains currently identified Roles:

3.1.1 Meeting Roles

Chair

Performed by the Team Leader and is responsible for running/controlling the meeting as well as deter- mining and distributing the agenda for the meeting. Apologies and items to be added to the agenda from other team members are to be sent to the Chair prior to meeting.

Sargent at Arms

Keeps an eye on proceedings and time spent on each item. Responsible for keeping the meeting on topic and on time. Their conduct must ensure that team members still have the ability to voice their opinions and ideas. Additional time can be utilized for discussion outside of the arranged meeting time.

Scribe

Records the meeting minutes and is responsible for circulation of meeting minutes. This role will be rotated through team members on a monthly basis as follows: Justin, Ben, Ha, Alex, Jason and Simon. The Chair is not expected to take minutes.

3.1.2 Formal Review Meeting Roles

Moderator

Plans the review and coordinates the review process

Scribe

Documents any issues or problems found during the review

Inspector

Examines the document/product for defects

Author

The creator of the work being reviewed

Reader

Reads out documents for the inspectors.

3.1.3 Champion Roles

Champion is a role that directly responsible for ensuring the quality of a particular area assigned to them, as well as the compliance of standards and procedures of their section's of the project. They are not however responsible for completing majority of the works. They are to delegate work, assist where appropriate and are the single point of contact for issues.

Team Leader is responsible ensuring the team works effectively together to achieve successful completion of the project. The Team Leader typically chairs team meetings and is the supervisor liaison. They should be notified of any political issues within the team. The team leader is also responsible for monitoring the completion of work logs to ensure members are recording time spend, and contributing. They are also responsible for writing status reports where applicable and attending team leader meetings.

Documentation Champion in in charge of making sure all documentation is consistent, complete and to a high standard. This includes Latex formatting and filenames.

SVN Champion is responsible for ensuring that the repository is used to its full potential and enforcing the SVN standards. They are also responsible for maintaining directory structure and resolving any issues with the repository.

Mantis Champion is responsible for maintaining the Mantis bugtracker and ensuring tasks are closed out once completed. They should encourage members to complete their allocated tasks.

Usability Champion needs to ensure the product meets usability related non-functional requirements. They should understand the Client Requirements and ensure that any products are built with the client use in mind. They should assist with testing to ensure the GUIs are usable and intuitive.

The Code Champion is responsible for quality and on schedule delivery of code. They are to ensure other members are building and uploading code that compiles and is up to standard. They are responsible for ensuring delegated works are completed in a timely manner.

Testing Champion is responsible for running and reporting on test results. This includes Unit and functional testing. They should work with the Usability Champion to ensure that exceptions and errors

are understandable and informative. They are to ensure tests carried out by other members are captured and recorded

3.1.4 Communication Roles

Client

The Client Liaison acts as the single point of contact between the team and the client. This allows coordination of all incoming and outgoing correspondence with the client and distribution to all team members. They are also responsible for setting up regular Client Meetings

Supervisor

The Supervisor Liaison allows the University and Supervisor to have a single point of contact for the team. However, other team members may contact the Supervisor for issues themselves. The role is typically filled by the Team Leader.

3.2 Tasks and Responsibilities

3.2.1 General Team Member Responsibilities

- If a team member is selected for a task they will complete the task by the allocated time. If unable to complete task in time, member is to raise an issue prior to deadline with the team leader.
- Meeting Actions are binding unless changed at a later meeting.
- Team members are responsible for the logging of their own time sheets.
- Members are to conduct themselves in an appropriate manner facilitating a healthy work environ- ment.
- Members are required to maintain communication with team.
- Members are required to follow all processes as described in the SQAP.
- Members must make their best effort attend all allocated meetings/workshops and are to submit an apology if they are unable to attend.
- Members are to follow all directives from champions.
- Members are to actively partake in group discussion and provide input to the product and the process.

3.2.2 Champions

Team Leader

- Responsible for the running of weekly team meetings.
- Responsible for the booking of weekly meeting room.
- Responsible for maintaining administrative documentation.
- Responsible for motivating and tracking team progress.
- Responsible for monitoring work logs and time spent on project.
- Responsible for being a point of contact for issues/resolution.

- Responsible for liaising with supervisor

Client Liaison

- This champion is directly responsible for all communications with the client.
- Correspondence from team members must be relayed to/from client in a timely manner via client liaison.
- Minor/non-urgent communications are to be collated to avoid bombarding client.
- Any compiled versions that need to be tested will be sent via the liaison.
- Liaison is responsible for ensuring client receives all relevant information for a test, as well as distribution of the test results supplied by the client to the team (bidirectional communication).

Usability and Graphical User Interface (GUI)

- Responsible for ensuring team members take usability into account during development.
- Responsible for ensuring consistency of GUI
- Responsible for ensuring that performance does not negatively impact usability.

Documentation

- Responsible for creating and maintaining document templates.
- Maintaining quality and standards of documents.
- Responsible for providing assistance with documentation issues.
- Responsible for maintaining documentation tools.

Code

- Responsible for quality control of code artifacts.
- Responsible for tracking code progress
- In charge of organising developer meetings to discuss progress and address any difficulties or con- cerns.
- In charge of ensuring appropriate workload is assigned for each developer.
- In charge of ensuring standards and best practices are met and followed, respectively, during the development process.

SVN

- Creating and maintaining repository.
- Monitoring commit messages.
- Maintaining file structure and location standards.
- Providing assistance with branching and merging.

Mantis

- Creating and maintaining Mantis
- Closing out resolved issue
- Monitoring progress of members with issues/tasks

Testing

- Completing or delegating running of tests
- Reporting results of tests to team and on SVN
- Building test documents
- Encouraging testing withing the team and compile results

Chapter 4

Documentation

4.1 Software Documents

4.1.1 SQAP

The SQAP is a plan written before any development that outlines all standard practices and procedures to ensure a quality process therefore help produce a high quality product.

4.1.2 SRS

A Software Requirements Specification will be developed to describe the behaviour of the proposed DMS as derived from client requirements. The SRS will be based on the IEEE 830 standard, but will be modified to make it appropriate for our project.
A general outline of the document is as follows:

1. Introduction
   (a) Purpose - Outline of the SRS
   (b) Scope - What the DMS is, what it will do and its application
   (c) Definitions, acronyms, and abbreviations
   (d) References
   (e) Overview
2. Overall description
   (a) Product perspective - How the product works with other tools, i.e dataflow
   (b) Product functions
   (c) User characteristics - Who will use the product, what training will they get
   (d) Constraints
   (e) Assumptions and dependencies
3. Specific requirements

4.1.3 Project Plan

A document to guide the building of the product. It will include a brief description of the project and why it should be built, what needs to be done to build the software and a timeline for when modules should be complete.

As more modules are mapped and more details are known about each item, the project plan is to be updated. It should also contain milestones where applicable and deliverable dates.

4.1.4 Module Plan

Module plan is to be developed following requirement analysis phase of each iteration of the software life-cycle.

Module plan contains:

- Module scope
- Design solution for module
- Work division
- Timeline of artifacts and testing

4.1.5 Self-assessment reports

A self-assessment report is to be completed by each team member each as per the unit outline that will provide evidence of work completed and self reflection. It will document knowledge and experience that has been gained during the process.

This document should contain the following sections

- Summary
- Work completed
- Mistakes made
- Knowledge gained
- Evidence

4.1.6 Audit Report

Whenever an audit is carried out a document must be produced that indicates the outcome; anything that does not follow the processes outlined in this document SQAP, whether the process is followed and corrective actions.

Audits can be carried out internally and externally.

4.2 Management Documents

4.2.1 Meeting Agendas

- This document will be of Latex type and will be prepared by the Team Leader (Or Meeting Chair) prior to each meeting.
- All team members are expected to contribute to the agenda by requesting topics of their choice be added to the agenda.
- Topics shall be owned by the team member who submitted them, unless otherwise stated.
- Owners shall attend the meeting prepared to introduce and discuss their topic.
- Submissions will be accepted by Close of Business (5:00 PM) (COB) the day prior to the meeting.
- The author of the agenda will upload to the SVN by COB the day before the meeting. A reminder email will also be sent to the required attendees with a PDF attachment of the agenda.

4.2.2 Meeting Minutes

- Will be collected at every meeting.
- Must follow the minutes template as outlined on the SVN.
- Will be collected as either a raw .txt file or LaTeX file and then converted to latex format.
- Formatted minutes will be released on the SVN no later than the following day's COB.
- A PDF copy of the minutes can also be emailed if requested, however, members are expected to find the minutes on the SVN and complete their actions independently.

Chapter 5

Standards, practices, conventions and metrics

5.1 Purpose

Standards are essential for measuring and thus ensuring software quality. This section covers technical, documentation as well as process standards which guide the project's development and management. These standards mainly govern the output quality of each project's deliverable: libraries, applications, documents. In addition, they also serve as the development guidelines throughout the projects.

This section also includes practices that the development team shall adhere to, and will be assessed against.

These standards and practices are the basis to ensure and measure quality of the project's deliverables. The detailed procedure for assessment against the standards and practices can be found in the Reviews and Audits section.

All documentation (including the full SVN repository) will be available for the client a pon completion of the project for any future usage.

5.2 Standards

The following standards will be used as the basis for quality control in this project. They shall be adhered to closely throughout the software life cycle. Standards will be reviewed to ensure that they are being met.

5.2.1 Coding Standard

The following language-specific standards are used
-Swinburne C# and .NET Standard
-Swinburne Java Standards (Spikes only, N/A for final coding as C# is the chosen language.)

5.2.2 Documentation Formatting Standard

Where possible, all text documents will be written in LaTeX to ensure they are easily merge-able.

- No non-standard LaTeX package should be used.

- The hyperref package should be used in all documents; so that all links function correctly.
- The line \setlength{\parskip}{1} can be added to the document header to remove paragraph indents, and use vertical space to separate paragraphs instead.
- All documents should use the fullpage package so that the margins are a reasonable and consistent size.
- No references should be hard-coded but rather use the \label-\ref standard.
- Only the document classes article, report and beamer should be used.
- No customizing of the fonts; everything should be left at the default.
- Tabs should be converted to spaces with a width of 4.
- Indenting of the mark-up should be done to improve readability.
- Figures:
  - All figures need to be in directory called "figures" where the TEX file is located.
  - All figures need to be wrapped in a \begin{figure}[H] command and labelled with an appro- priate caption under the figure.

```
\begin{figure}[H]
        \centering
        \includegraphics{imagefile.png}
        \caption{This is the caption}
\end{figure}
```

- Tables
  - All tables need to be wrapped in a \begin{table}[H].
  - All tables should be labeled with an appropriate caption under the table.
  - All tables should have borders like in the example below
  - The use of long tables is accepted to allow tables to span multiple pages.

```
\begin{table}[H]
        \centering
        \begin{tabular}{|l|l|l|}
            \hline
            col1 & col2 & col3 \\ \hline \hline
            1    & 2    & 3    \\ \hline
        \end{tabular}
        \caption{This is the caption}
\end{table}
```

- Acronyms should be added to an acronyms table created with the code example below, and used with the \ac{} command

```
\section*{Acronyms}
\begin{acronym}[TDMA]
        \acro{COB}{Close of Business}
    \acro{IEEE}{Institute of Electrical and Electronics Engineers}
\end{acronym}
```

- Presentations should be done using the LaTeX beamer package. The Warsaw theme should also be used to ensure visual consistency.

5.2.3 Filename/Location standards

- All file and folder names will be lowercase. With the exception of the code folder where uppercase characters are allowed for the purposes of integrating with Microsoft Visual Studio's standards.
- There shall be no whitespace (spaces) in filenames.
- " _ " will be used to delimit the file and folder names in the event that the name is multiple words.
- Management/Administration files will be named as follows "filename yyyy mm dd".
- Coding files shall be organised in folders in this structure \trunk\code\eagle\subproject\, \subproject is the programming unit (Java or .NET projects)
- Coding file shall be named in format of subprojectabbr namespace filename.extension, where names- pace represents the subcomponents of a sub project.
- Management/Administration related files are to kept within the docs folder.
- Multiple related files with similar content such as the meeting minutes are to be stored within an appropriately named encapsulating folder.

Document Tree

The following document tree describes the SVN structure. Additional folders may be added at the discretion of team members after consulting with the SVN champion.

```
repos
      \trunk
          docs
               meetingsminu
                    tesagend
                    as
               presentations
                    project_presentation_1
                         figures
                sqap
                    figures
                    references
                    release
               project_planfi
                    gures
                    release
               self_assessment_reports
               srs
                    figuresr
                    e l e a s e
                    mockup
               standardsw
               orklogs
          code
               eagle
                    subproject
      \tags

      \branches
```

5.2.4 SVN standards

- All Commits to the SVN are required to have a corresponding message, which follows the following standard:

  [Mantis Issue ID] file.ext – Section/Module – List of Changes (brief and concise).

  In the event that sections are not relevant the fields are still included but left blank. i.e. [] file.ext - - committed the new file
- Ensure files to be committed are not currently open in their respective editors.

Branching Standard

This SVN will utilise the Branch-When-Needed system. Users

commit their day-to-day work on /trunk.

1. /trunk must compile and pass regression tests at all times.
2. A single commit (change-set) must not be so large so as to discourage peer-review.
3. If rules 1 and 2 come into conflict (i.e. it's impossible to make a series of small commits without disrupting the trunk), then the user should create a branch and commit a series of smaller change- sets there. This allows peer-review without disrupting the stability of /trunk.

Pros: /trunk is guaranteed to be stable at all times. The hassle of branching/merging is somewhat rare.

Cons: Adds a bit of burden to users' daily work: they must compile and test before every commit.

5.2.5 Document Releases

In the event that a document is released to an outside party; be it submission to the university or the client it must be:

- Converted into a static format such as a pdf
- Appropriately renamed with a revision number
- Moved to a release folder within its current folder

Each release will be named as follows: filename rxxx, where xxx is a number. The first release will be 100 and each additional release will be incremented by 10 i.e. filename r100 will be followed by filename r110.

5.3 Practices

The following practices will be used as the basis for quality control in this project. They shall be adhered to closely throughout the software life cycle. Practices will be audited to ensure that they are being followed appropriately.

5.3.1 Communication Practices

Client

- All contact will be made to the client through the client liaison champion.

- In the event that the champion is absent or on leave, alternative arrangements will be made (In advance if possible).
- Contact will be primarily made through email.
- Phone calls may be used in the event that emails can not provide enough detailed information.
- Meetings with the client shall be regular, taking place approximately every three weeks. However correspondence via phone or email should be more regular
- Meetings will always have a minimum of two team members present.

Team

- Email will be the primary method of communication, when face to face communication can not be conducted.
- Student email addresses are to be used in all email communications.
- Emailing of documents will not occur if the item can or is held in the SVN with the exception of finalised meeting agendas and meeting minutes in PDF format only.
- Emails need to be checked daily.
- If a request for reply or acknowledgement of email receipt, members are to respond upon reading for the first time.
- Email communication will be kept to a professional standard.
- Emails can be used to confirm verbal contracts.
- Skype meetings are permitted if face-to-face meetings are unable to be arranged, but should be kept to a minimum.

Supervisor

- Formal contact with the supervisor will conducted by the Team Leader.
- Supervisor will be present at one meeting per fortnight, at the request of the team leader (typically the Weekly Team Meeting).
- Contact will primarily be through email.
- Agreements with supervisor will be confirmed via email.
- All emails to the supervisor should CC the entire team unless they are of a personal nature.
- If the team meeting is held out of bussiness hours, then a seperate meeting should be arranged with the supervisor. If this cannot be achieved, then at the very least a status report should be emailed to the supervisor to ensure they remain up to date with the progress.

5.3.2 Meetings

- Team meetings will be held on a weekly basis and will have a standard length 30 to 60 minutes.
- All team members are required to be present.
- If a team member can not be present, an apology needs to be communicated directly to the team leader As Soon as Possible (ASAP).
- All meetings require minutes to be taken.

- A meeting outside of the weekly team meeting does not need all members present; notes are required.
- Any meeting with the client requires at least two members present, again notes will be taken. Notes will distributed to client for confirmation.
- notes will be stored on the SVN

5.3.3 Worklogs

- Weekly update of Worklogs on the SVN
- Team Leader to monitor Worklogs.
- Team Leader to monitor and maintain the project hours summary sheet

5.3.4 SVN

- Temporary/intermediate files are not to be committed

5.3.5 Coding practices

General guidelines

- Strictly follow C# and .NET standards as outlined in 5.2.1
- Keep the code simple, avoid using unnecessary "clever" code.
- All source files must use the standard header template
- All methods, fields and properties must have comments that follows the official standard in 5.2.1 , inherited class only requires comment if behaviors differ greatly from that outlines by its parent classes
- Public properties are acceptable, however default setters are discouraged, and guards should be used in writing setters.
- Development must be in-line with architectural design, in order to ensure transparency in code.
- Once a week meetings (15 minutes) to report progress/difficulties in development, could be con- ducted after weekly team meeting.The aim is to ensure problems are known early and progress are understood by the whole team.

Guidelines on project structure

There shall be only one Windows Form Application project for a particular module, it shall be the only User Interface project.

All major components (ReportGenerator, DatabaseIO etc.) are to be implemented by C# libraries.

User interface is to be implemented by native .NET form creator to ease future development, XML-based interface design is discouraged.

There shall be a library project called EagleStaticCommon which contains helper or data components that are used by all other components. To avoid breaking the layered architecture, EagleStaticCommon shall only contain static functions and properties. However the decision of putting a component in EagleStaticCommon must be considered carefully to avoid overcrowding, which lead to confusion.

Guideline on components design

Each component design must be justified by thorough analysis into quality requirements of the compo- nent, applied design patterns or tactics.

It is recommended that to the very least, a component design should incorporate and separate the following component into directories in the library:

- Interface: include all abstract classes and interfaces
- Enumeration: all enumerations
- EventArgs: all subclasses of EventArgs if any
- Structures: all data structures if any
- Contracts: contract classes if any

Guideline on naming and namespaces

Appropriate namespaces will be automatically created by Visual Studio if folder structure is set up in the project.

Chapter 6

Reviews and Audits

6.1 Purpose

This section of the SQAP defines a set of procedures used to validate project deliverables and to verify team processes with respect to defined requirements and standards.

The purpose of validation is to ensure that the correct deliverables are being produced with respect to the client requirements and team standards. This is done through internal and external reviews.

The purpose of verification is the ensure that processes outlined in the SQAP are followed to ensure product quality. This is done through internal and external audits.

The standards, procedures and practices can be found in chapter 5, Standards and Practices.

6.2 Review/Audit list

6.2.1 Reviews

Reviews are held during all phases of the project's life-cycle.

Formal Review Process

All formal review meetings must use the following process, a formal review is to be declared on a case by case basis:

1. A review committee is selected and the specified roles are filled.
2. The Moderator identifies and/or confirms the review's objectives.
3. The Moderator ensures that all members of the committee understand the objectives and the review process.
4. (a) Individual: the review committee will prepare to review the work by examining it carefully for potential defects.

   (b) Team: the review committee meets at a planned time to pool the results of their preparation activity and arrive at a consensus regarding the status of the document or standard being reviewed
5. Author of the work makes the required changes as specified by the review committee.
6. Moderator verifies that the actions required by the Author have taken place.

Informal Review Processes

Code

Code quality is to be ensured through regular reviews as listed below. In the event that code is found to be unsatisfactory the results will be communicated to the relevant team member and raised as an Issue.

1. Peer review: Code commits shall be reviewed by a peer developer, assigned in the team meeting, as recommended by the Code champion, for the following aspects. Weekly inspection will be carried out on all commits by the assigned peer prior to the next meeting.
    - Coding Standard
    - Task Completion
    - Verified against initial specifications, from each stage's detail design.
    - Agreement upon any changes to specifications
2. Client review: Every 2 weeks, all working branches are merged and sent to client for testing and review against the following: (The method of transfer will not require meeting with the client face to face and is distinctly different from client meetings.)
    - Deliverable time line
    - Verified against specifications
    - Validate task completion

Meeting

Meeting quality will primarily be maintained through audits of the correct process but all meeting related documents will also be reviewed for quality.

Meeting minutes will be reviewed following the first meeting of each secretary against the standards. This is done alongside the formal acceptance of minutes at the conclusion of each meeting.

Agenda will be accepted prior to each meeting and formally reviewed prior to the following meet- ing.

Any documents found to be unsatisfactory will have results communicated to the secretary and raised as an Issue.

Management Document

Management documents will be reviewed against document standards prior to being finalised and re- leased. In the event that the document is found to be unsatisfactory, a list of improvements will be generated and raised as an issue. One example of this is the feed back sheets provided by the supervi- sor.

6.2.2 Audits

Audits should be held regularly during all phases of the project's life-cycle to ensure processes put in place are being adhered to.

Coding Practices Audit

Coding practices will be audited by code champion on a case by case basis (normally as a result of consecutive unsatisfactory peer reviews). Failure to meet defined coding processes will result in a list of improvements being generated and communicated to responsible team member/s.

Communication Audit

Communications will be audited on a monthly basis by Team Leader. Failure to meet defined communi- cation processes will result in a list of improvements being generated and communicated to responsible team member/s.

SVN Practices Audit

SVN Practices will be audited as part of the routine maintenance by the SVN champion. Failure to meet the prescribed practices will be communicated to relevant team members with recommendations for improvement.

Chapter 7

Testing

Testing will be predominately in the form of field testing with the assistance of the client. This is the accepted method due the domain knowledge requirement of the user, and the restrictive nature of the project. The team are to perform as many tests on the releases before handing over to the client. Unit tests can be utilised, but will not make up a large part of the testing. The can be used for confirming numerical data generated by the program has been manipulated correctly.

The team will also perform some usability/function testing on the prototype prior to release. This will most likely be involve trying to break/crash the program. As the project is heavily reliant on import and export of data, there are numerous times when IO Exceptions could be thrown. As part of this testing, the tester is to document all error, messages and general comments relating to usability of the GUI. The tester should also be provided with a list of performance metrics to be tested and monitored during the testing process. Some examples are: CPU usage, memory usage, processing time, closing opened files and termination of processes correctly.

Once the team has made their best effort to break the program, handled all exceptions, provided usable error messages and are satisfied with the release, it is to be given to the client for feedback.

The release should be accompanied with feedback sheets for the client to complete and return to the team so that issues can be repeated, and ultimately, resolved. The client will also be expected to provide an acceptable performance level for the metrics used. For example what is the acceptable processing time; 1 minute or 1 hour?

7.1 Requirement

The overall goal of the team is to satisfy the clients needs when building the software. Therefore, strong consultation with them will result in a product that works, is usable and maintainable.The requirements outlined in SRS shall be verified and validated by the client to ensure the product is suitable for deployment and use.

7.2 Use case generation

Use cases shall be validated and verified by the client with the assistance of the testing team. Sample outputs from the client will allow a basic understanding for the team, but ultimately the client will be responsible for communication specific uses of the software to ensure the team can adapt it to suit.

7.3 Installation and User Documentation Generation

The releases will be an in the form of an executable and a set of .dll files that will be provided to the client. There is no installation as such, the user simply runs the executable. All input and output files will be located as part of the GUI.

The client will also be supplied with a comprehensive user document that details the GUI and should have some examples of how to use it. The client will also be supplied documentation of the code and the design documents. This is to assist with maintainability and use for future projects.

Once code has been finalised, and testing is completed, they will be supplied with all of the source code and a final build of the software.

- Tailored for a particular module, tests will be outlined within the module plans.
- Modules will be tested against a set of defined use-cases as agreed by client.
- Testing will consist of both black and white box testing.

Chapter 8

Problem reporting and corrective action

8.1 Personnel

If an issue arises with personnel the Team Leader is to be notified. The Team Leader will follow up on the issue to gather all the facts. Once known the Team Leader will suggest a corrective action. Corrective Action can include but is not limited to: counselling, team reorganization, protocol changes.

8.2 Work

8.2.1 Project major time line

There is one parent project, which is ProjectEagle. ProjectEagle shall contain multiple sub-projects, which are the major modules that will be developed: Module 1-2, Module 3, and Module 4.

ProjectEagle will have have multiple Versions (Manage \Project \Version), each shall be a major mile- stone for development.

Each sub-project will also have multiple version, with each major version is one iteration.

To implement spiral process model, the following postfix is included into each sub-project version to indicate the stage

- " dsg" indicating the design stage.
- " dev" indicating the development stage.
- " tst" indicating the testing stage.

Release date of each version is the due date, scope is to be planned according to this due date.

Progress of the current stage can be viewed by visiting "Road map" page.

8.2.2 Stage-dependent tasks

Stage-dependent issues and tasks must indicate which sub-project they belong to, as well as which version (e.g. [Module 1-2] v1.0 dsg indicating this issue belong to design phase of Module 1-2 first release.

8.2.3 Crossed-states tasks

Crossed-states tasks and issues must indicate parent's version. For example, documentation tasks may fall under ProjectEagle v1.0.

8.2.4 Task creation

Minute taker is to convert meeting's actions to task and assign to appropriate developer.

Developer is responsible to divide the task logically into smaller tasks if necessary.

Team member would also create task as appropriate: for bug reporting or planning. Task

creator must check for existing issue prior to creating task.

8.2.5 Task assignment

When a task cannot be assigned upon creation, the respective champion of the task must perform assignment within 24h of task creation.

8.2.6 Task life

Assignee (who is assigned to the task), must response within 12h if the assignment is deemed inappro- priate.

Resolver is responsible to ensure solutions are checked against the appropriate standards and practices prior to marking the issue as "Resolved".

It is stressed that the resolver must entered the time spent on the task into the time-spent box before confirming as 'resolved'

After an issue is marked "Resolved", respective champion is responsible to formally/informally review the task (exception for trivial tasks), then mark the issue as "closed"

8.2.7 Issue Categories

Categories can be updated to adapt to the project's development, the following are most up to date:

- Administration
- Audit - External
- Audit - Internal
- Client Liaison
- Coding - Prototype
- Documentation - AD
- Documentation - General
- Documentation - PP
- Documentation - SAR
- Documentation - SD
- Documentation - SQAP
- Documentation - SRS

- Lecture
- Meeting - Client
- Meeting - Other
- Meeting - Weekly
- Presentation Preparation
- Research - Coding
- Research - Documentation
- Review - External
- Review - Internal
- SVN Management
- Requirements - Module 1
- Design - Module 1
- Coding - Module 1
- Testing - Module 1
- Requirements - Module 3
- Design - Module 3
- Coding - Module 3
- Testing - Module 3

Chapter 9

Tools and methodologies

9.1 Tools

9.1.1 LaTeX

To compile the documentation TEX files, pdflatex should be used, on Windows the MiKTeX 2.9 package provides this. The recommended editor for LaTeX files is the Texmaker editor, version 3.3.1 has been recommended.

9.1.2 Issues tracking

Issue tracking tool is Mantis. The tool can be access by SIMS username and password at:
https://mercury.it.swin.edu.au/hit3158/hit3158_02/mantis/Mantis provides a simple to use and feature-rich platform to track issues, progress and effort by the team.

9.1.3 SVN

All commit and updates by team members shall be done through TortoiseSVN or SmartSVN client to avoid potential problems. Both programs have been tested to work, so it is up to the individual to choose which program they prefer.

9.1.4 Visual Studio

Visual Studio 2010 (available via MSDN Academic Alliance in association with Swinburne FICT faculty) is selected as the standard C# development environment. It provides state-of-the-art development tool as well as supporting client's .NET 2.0 standard environment.

9.1.5 Virtual Machine

All development will be done on a virtual machine with a Windows XP 32bit image provided by the client. The recommended Virtual Machine software is VirtualBox version 4.1.8.

9.1.6 Skype

If Skype meetings are deemed to be necessary, then all team members will need to download and install Skype and have access to a microphone and speakers.

9.2 Design Methodology

- Spiral Lifecycle Model
- Lifecycle model utilises multiple iterations which suits our modular design.
- Each iteration will have its timeline specified in the project plan.
- Each phase will have its timeline define in the module plan.
- Can have independent modules being processed concurrently if practical.
- The spiral model is endorsed by the client, and the shared understanding of the methodology will assist in communication on the topic of project planning.
- The model treats each iteration as a new project plan modified to include the following phases:
  - Requirements: Determining requirements for current iteration of the project
  - Design: Building a project plan for this iteration including timeline, work allocation, and risk identification and management.
  - Develop/Test: Programming, prototyping, documenting, verifying and validating as discussed in Design Phase

This modified model can be seen in the figure below.

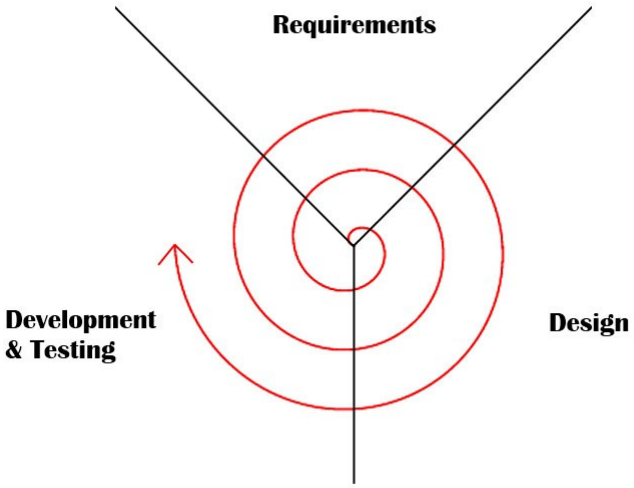

Figure 9.1: Spiral Software Lifecycle Model

Chapter 10

Records collection, maintenance and retention

Minutes, Agendas and Notes from meetings are added to project team's SVN Repository as described in SVN Procedures. Minutes and Notes will be added following approval by meeting participants.

All documentation will be retained in repository for the duration of the project.

Chapter 11

Risk Management

11.1 Purpose

Risk management is undertaken to facilitate the creation of a product that is high quality, on time and delivers the scope specified by the client.

11.2 Categorization

For this project three major categories of risks have been identified:

1. Risks with respect to the work to be done.
2. Risks with respect to the management.
3. Risks with respect to the client.

In the following sections each of these categories have their major risks identified. For each risk, a description, a probability to occur, its impact and the preventative/(reductive) action associated are given.

Both the probability of a risk occurring and the impact of a risk if it does occur have been quantified as being low, moderate or high. Actions have been categorized as preventative and reductive; preventative actions aim to reduce the likelihood of risks occurring and reductive actions reduce the impact of risks if they do occur.

11.3 Risks with respect to the work to be done

- Corruption of repository
  - Probability: Low.
  - Impact: High resulting in loss of work.
  - Reductive Action: Weekly backups plus local checkouts reduce impact significantly.
- Design Errors
  - Probability: High.
  - Impact: High, design errors would potentially increase production time and/or produce a deliverable not valid to client requirements.

  - Preventative Action: Rigorous design methodology prior to development.
- Time Shortage
  - Probability: High.
  - Impact: High, resulting in a loss of product quality, loss of functionality or delivered past deadline.
  - Preventative Action: Rigorous design methodology prior to development including work dis- tribution and conservative timelines.
- Illness or absence of team members
  - Probability: High.
  - Impact: Variable impact dependant on time in schedule.
  - Reductive Action: Shared understanding of work allows load to be distributed.
- Software non deployable
  - Probability: Moderate.
  - Impact: High, will be unable to provide client with the DMS.
  - Preventative Action: Regular contact with client with minor releases to ensure that they can be deployed on the system.

11.4 Risks with respect to the management

- Illness or sudden absence of team leader
  - Probability: Moderate.
  - Impact: Variable impact dependent on time in schedule.
  - Reductive Action: Emergency meeting to be organised by Team Leader to elect temporary team leader.
- SQAP not suitable for our purposes
  - Probability: Low.
  - Impact: High, failure to follow SQAP would reduce product quality.
  - Preventative Action: SQAP to be produced with full team input and cleared with team supervisor.
- Team member leaves the team
  - Probability: Low.
  - Impact: High, their roles and responsibilities are no longer being fulfilled, and the amount of manhours the team can provide in a given time is diminished.
  - Preventative Action: None. However, roles and responsibilities will need to be redistributed to the remaining team members.

11.5 Risks with respect to the client

- Changing client requirements
  - Probability: Moderate.
  - Impact: Moderate, increased workload and timeline issues.
  - Preventative Action: Rigorous discussion of requirements plus official SRS document early in project timeline.
- Client unavailable
  - Probability: Moderate.
  - Impact: Low, unavailability for questions and software releases
  - Reductive Action: Vital questions for client to be communicated before they become critical.
- Client abandons project
  - Probability: Low.
  - Impact: High, No more work modules for the project to complete.
  - Reductive Action: Get as many modules and requirements off the client as possible.