Software Quality Assurance Plan

Swinbots - Swinburne RoboCup Team
SEP Group 1, 2004

Author(s):	Andrew Walker - 1262106
	James Fraser - 1261371
	Shane Harvie - 1262246
	David Reed - 1266659
	Luke Pitcher - 1262769
	Nickolas Wanke - 1229338
	Dejan Zigic - 1265636
	Michael Henderson - 1261487
Last Modified:	October 4, 2004
Version:	*Revision* : 1.32

Modification History Table

Date	Version	Description	Author
24/02/04	1.1	Initial document	AW
26/02/04	1.2	Spelling/Grammatical Editing, further content	JF
03/03/04	1.3	QA Roles/Responsibilities	JF/SH/LP
09/03/04	1.4	Many editing changes	AW
13/03/04	1.5	Organisation, Review Work Flow Diagram	SH
13/03/04	1.6	Development Process	SH
13/03/04	1.9	Added several sections from BSE checklist	SH
13/03/04	1.10	Added content for Tasks section	MH
13/03/04	1.11	Made bullet points consistant	AW
16/03/04	1.12	Changed to make directory name in CVS table	AW
21/03/04	1.13	Addition of communication standards, timelog standards	SH
22/03/04	1.14	Removed physics simulator related material, added Tactics Debugging info	SH
22/03/04	1.15	Added to email standard, added Test Report Evaluation as per review f/b	SH
22/03/04	1.16	Fixed broken build	AW
22/03/04	1.17	Added template for Agendas and minutes as per review f/b	SH
23/03/04	1.18	Changed placement of review sections as per review f/b	SH
26/03/04	1.19	Removed DevProcesses diagram and referred it to SDLC model diagram in PP	SH
28/03/04	1.20	Added List of tables and List of Figures as per review f/b	SH
28/03/04	1.21	Added document overview section	SH
28/03/04	1.22	Minor edits	JF
05/04/04	1.23	Altered review procedure	AW/SH
22/04/04	1.24	Corrected CVS table, added note about testing procedure	AW
06/05/04	1.25	Added 3 blank lines before Intro. and updated history for previous 3 revisions	JF
24/06/04	1.26	changed meeting minutes filename format to suit new type	SH
05/07/04	1.27	Edited several sections, refer to sqap_review_dz in docs_review folder	DZ
01/10/04	1.28	Added "After Release Changes" section (empty)	SH
02/10/04	1.29	Added abstract and intended audience	SH
02/10/04	1.30	Added "After Release" changes. Minor editing.	SH
02/10/04	1.30	Added "After Release" changes. Corrected spelling mistakes.	AW

Abstract

This document presents the Software Quality Assurance Plan for the 2004 RoboCup team. It defines the processes that will be undertaken and products that will be produced, to ensure that a high quality project is completed on time, within budget, and with all scope requirements fulfilled. It describes the management processes that will be undertaken and the team structure that will adopted in order to fulfill our quality needs. Most importantly, it describes the standards, practices, conventions and metrics that have been defined in order to produce a quality final product. This includes review processes and relevant document templates.

It is envisaged that through adherance to the processes defined in this document, software modules will be produced in an efficient manner, and testing of these modules will be thorough, and well controlled. A successful project will result.

Audience

The target audience for this document is:

- The Client — Dr Ali Bab-Hadiashar
- Project Team Members
- Project Supervisors
- Past and Future Team Members

Contents

1 Introduction **1**

1.1 Purpose . . . 1
1.2 Scope . . . 1
1.3 Project Justification . . . 1
1.4 Deliverables . . . 1
1.5 Definitions . . . 2
1.6 Acronyms . . . 2
1.7 References . . . 2
1.8 Document Overview . . . 3

2 Management **3**

2.1 Organisation . . . 3
2.2 Quality in the Software Development Life Cycle . . . 4
2.3 Tasks . . . 4
2.4 Responsibilities . . . 5
2.5 Major Checkpoints and Milestones . . . 5

3 Documentation **5**

3.1 Project Plan . . . 6
3.2 Software Quality Assurance Plan . . . 6
3.3 Software Requirements Specification . . . 6
3.4 Software Design Document . . . 6
3.5 User Task Analysis . . . 6
3.6 Software Test Plan . . . 6
3.7 Risk Management Plan . . . 6
3.8 Software Configuration Management Plan . . . 6
3.9 Project Progress Report . . . 7
3.10 User Manual . . . 7
3.11 Additional Documents . . . 7

4 Standards, Practices, Conventions and Metrics **7**

4.1 Meeting Procedures . . . 7
4.1.1 Meeting Attendance . . . 7
4.1.2 Calling a Meeting . . . 7
4.1.3 Meeting Procedure . . . 7
4.1.4 Meeting Agenda Format . . . 7
4.1.5 Meeting Minutes Format . . . 8
4.2 Documentation Standards . . . 9
4.3 Document Naming Standards . . . 9
4.4 Coding Standards . . . 10
4.5 Comment Standards . . . 10
4.6 Testing Standards . . . 10
4.7 Timelog Standard . . . 10
4.8 Communications Standards . . . 11
4.8.1 E-Mail Standards . . . 11
4.8.2 Out-of-office standards . . . 11
4.8.3 Meeting Agenda Communication . . . 11
4.8.4 Written Communication Standards . . . 11
4.9 Metrics and Measures . . . 11
4.10 Compliance Monitoring . . . 12

5 Review **12**
5.1 Review Procedure 12
5.1.1 Before the Review Meeting 13
5.1.2 During the Review 13
5.1.3 After the Review 13

6 Testing **13**
6.1 Test Procedure 13
6.2 Evaluation of Test Reports 14
6.3 Client Validation 14

7 Problem reporting and corrective action **14**
7.1 Defect Reporting 14
7.2 Defect Correction 14
7.3 Conflict Resolution Process 14

8 Tools, Techniques and Methods **14**
8.1 Software Development Life Cycle 14
8.2 Choice of Language 14
8.3 Tools for Coding 15

9 Code Control **15**

10 Records Collection, Maintenance and Retention **15**

11 Training **15**

12 Risk Management **15**

Appendices **16**

A Required “After Release” Changes **16**

List of Figures

1 Steps to the Review Process 12

List of Tables

1 Directory Structure 16

1 Introduction

1.1 Purpose

The Software Quality Assurance program is established to ensure that software products developed as part of the RoboCup project meet all software-related requirements. This Software Quality Assurance Plan (SQAP) details Quality Assurance conventions, practices, standards and techniques to be employed by the RoboCup team.

This SQAP has been written in order to satisfy requirements internal and external to the project team. Internally, it forms the important first step in ensuring that all software products are developed in an efficient and effective manner. Externally, it provides the client with a firm understanding of the measures being taken to ensure that the final product meets their requirements to a high standard.

1.2 Scope

The software products to be developed under this quality plan are identified as:

- Image Processing
- Tactics Engine
- Embedded robot code
- Radio-Frequency Communication System
- Tactics Debugger

All of the above software products play an integral role in the success of the RoboCup project. Each component represents a 'module' of an overall software system, and as such, each component must be designed and implemented to operate seamlessly within this system.

This document has been been written in conformance with the IEEE's "Guide for Software Quality Assurance Planning" (based on ANSI/IEEE Std 730-1984, IEEE Standard for Software Quality Assurance Plans).

1.3 Project Justification

The RoboCup project is a chance for participating Robotics/CSSE students to apply the knowledge acquired during the previous four years of university study. RoboCup is an endeavour in various fields spanning the double-degree curriculum, with engineering activities including mechanical design, electrical/electronic design, software engineering and project management. The relatively large group size ensures that all members are able to test their ability to work within an organised group environment, whilst maintaining freedom to investigate ideas and engineering issues on an individual basis.

1.4 Deliverables

Refer to the 2004 RoboCup Project Plan, Section 4.1 Description of Deliverables.

1.5 Definitions

The definition of terms, acronyms and abbreviations.

CVS Repository (CVS) Data storage for all working files.

Change Management Formal process for the recording, analysing, estimating, tracking and reporting of changes to baseline project requirements.

Module (software module) A portion of a software system that performs a specific function and may be used alone or combined with other modules of the same system.

Hardware The physical components of the RoboCup system, e.g. chassis, wheels, microcontroller board.

Review Specific analysis of a part or parts of the system by logical, physical or testing means.

System The Software Package, including all relevant peripherals and documentation.

Mercury The Swinburne CVS/Development server (*mercury.it.swin.edu.au*).

1.6 Acronyms

AD	Architecture Document
BSE	Bachelor of Software Engineering
CS	Coding Standard
CSSE	Computer Science and Software Engineering
CSV	Comma Separated Values
CVS	Concurrent Versioning System
GUI	Graphical User Interface
IDE	Integrated Development Environment
PP	Project Plan
QA	Quality Assurance
SDD	Software Design Document
SDLC	Software Design Life Cycle
SEP	Software Engineering Project
SQAP	Software Quality Assurance Plan (i.e. this document)
SRS	Software Requirements Specification
TP	Test Plan
UTA	User Task Analysis
WinCVS	Windows GUI for the Concurrent Versioning System

1.7 References

- "SQAP Swinbots - Swinburne RoboCup Team" - SEP Group 2, 2003
- SQAP Template - http://wwwis.win.tue.nl:8080/2R690/sqap.html

1.8 Document Overview

This document — the Software Quality Assurance Plan (SQAP) — identifies the processes that will be undertaken in the RoboCup project, 2004, to ensure a quality final product. It has 12 major sections:

- **Introduction** (Section 1) provides an overview of the entire SQAP document, the project and the product being specified.
- **Management** (Section 2) describes the management processes that will be undertaken, and the team structure that will adopted, in order to fulfill our quality needs.
- **Documentation** (Section 3) outlines all the documentation that will be produced to ensure that a high quality project is completed on time, within budget, and with all scope requirements fulfilled.
- **Standards, Practices, Conventions and Metrics** (Section 4) describes the Standards, Practices, Conventions and Metrics that have been defined in order to produce a quality final product.
- **Review** (Section 5) describes the documentation and coding review processes that will be undertaken by the team.
- **Testing** (Section 6) describes the choices made with respect to testing to ensure a quality product.
- **Problem Reporting and Corrective Action** (Section 7) describes the methods for Defect Reporting and Correction, and the Conflict Resolution Process.
- **Tools, Techniques and Methods** (Section 8) describes the Tools, Techniques and Methods that will be used throughout the project to ensure a quality final product.
- **Code Control** (Section 9) describes in detail the methods and techniques that will be implemented to control the code produced throughout the project.
- **Records Collection, Maintenance and Retention** (Section 10) describes the methods that will be undertaken to ensure that important documentation is not discarded or misplaced.
- **Training** (Section 11) describes the training that will be undertaken to ensure that all team members are up to speed.
- **Risk Management** (Section 12) gives a brief overview of the Risk Management Plan, and how it relates to quality.

2 Management

2.1 Organisation

For general organisational structure refer to the 2004 RoboCup Project Plan Section 3 - Management organisation.

A Quality Assurance (QA) team must be formed to take ownership of all QA related activities. This team will be responsible for identifying and addressing any deviations from QA processes and standards specified in this SQAP. It will comprise a Code QA officer and a Documentation QA officer.

Documentation QA officer - James Fraser
Code QA officer - Andrew Walker

2.2 Quality in the Software Development Life Cycle

The SQAP plays an integral part in the Software Development Life Cycle Model, which is shown in the SDLC section of the Project Plan. Each of the steps have been chosen with the specific purpose of ensuring quality. They are as follows:

- *Project Planning:* Ensures that the project is completed on time and within budget, whilst accounting for all the steps required to ensure quality.
- *Definition of Standards:* Ensures that the project meets the required quality standards and is consistent throughout.
- *Literature Reviews:* Ensures that work completed by third parties is used in a manner to maximise the success of the 2004 RoboCup Team.
- *Requirements Definition:* A concentrated effort will be made during the initial phases of the SDLC model to define all requirements for all modules. This will aid in the prevention of scope creep, by defining all the requirements for the project before sign-off.
- *System Design:* Ensures that each of the system modules are integrated in such a way as to result in an efficient final product, of high quality.
- *Module Design:* Ensures that individual modules meet all requirements and can be integrated in an efficient manner.
- *Test Plan:* By defining the test plan before coding is commenced, the required performance of each of the modules, and the system as a whole, is explicitly considered. This will aid in the efficiency of implementation.
- *Implementation of Module Test Stubs:* By creating the Module Test Stubs before the actual modules are created, testing can be performed in parallel with coding of the modules. Once a module is thought to be complete, the tests can immediately be ran on the module. Any required changes are easily implemented, since the code is fresh in the developers mind. This will improve the efficiency and quality of the Module Implementation.
- *Prototype Construction:* Enables a check of the ability of the module design to meet the functional requirements.
- *Review of Prototype:* Allows the chance for a rethink about how the module design will meet the functional requirements, and the impact it will have on quality. Requirements can then be reviewed if required.
- *Module Implementation:* This is when the quality characteristics will be implemented in each of the modules.
- *System Implementation:* This is when the quality characteristics will be implemented in the final product.
- *Verification and Validation:* The final measure of quality. The system will be tested and improved to ensure a quality final product.

2.3 Tasks

The main QA tasks that will be performed are:

- *Informal Internal QA Reviews* - Quality issues may be raised and discussed at the weekly meetings. These issues will be resolved through discussion between the QA team and the appropriate team leaders.

- *Formal Technical Review Meetings* - See the *Review Procedure* section of this document.
- *Requirements Tracing* - Requirements Tracing ensures completeness of the software. A check is done to make sure that all lower level requirements have in fact been derived from higher level requirements, and that all higher level requirements have been allocated to lower level requirements.
- *Verification (including testing)* - See the *Testing* section of this document.
- *Metrics Collection and Analysis* - Various metrics will be collected and analysed using spreadsheets and MS Project. See the *Metrics* section of this document.

2.4 Responsibilities

The Code QA officer is responsible for:

- Reporting - weekly team meetings may have, as part of the agenda, a QA reporting item, as deemed necessary by the Code QA officer. The reporting item will only be added to the agenda when important QA issues are encountered regarding adherence to coding standards and correct usage of the CVS repository. Urgent QA issues may also be sent to all team members via Email. The QA officer and Team Leader must designate any code-quality rectification work during the meeting or specified in the email.
- Reviewing - The review process is outlined in the Work Flow Diagram in the *Review* section below.

  The Code QA officer will organise regular code reviews on a fortnightly basis. Review teams are to be formed by the Code QA officer with the input of the Team Leader, hence ensuring appropriate human resource allocation. The review process is described in the *Review* section below.

The Documentation QA officer is responsible for:

- Reporting - weekly team meetings may have, as part of the agenda, a QA reporting item, as deemed necessary by the Documentation QA officer. The reporting item will only be added to the agenda when important QA issues are encountered regarding adherence to documentation standards, during appropriate stages of the document writing process. Emails may also be used to make team members aware of any urgent QA issues in need of refinement. The QA officer and Team Leader must designate any documentation-quality rectification work during the meeting or specified in the email.
- Reviewing - the Documentation QA officer will organise documentation reviews prior to deadlines for deliverable documents, and in close proximity to scheduled completion dates for non-deliverables. Review teams are to be formed by the Documentation QA officer with the input of the Team Leader, hence ensuring appropriate human resource allocation. The review process is described in the *Review* section below.

2.5 Major Checkpoints and Milestones

For Major Checkpoints and Milestones, please refer to the RoboCup 2004 Project Schedule.

3 Documentation

The following subsections outline all the documents that will be produced throughout the development process. The team will undergo a formal review process of all documents, as outlined in figure 1. For some of the documents, a secondary review will be performed with a BSE student. Required improvements will be noted, and then implemented, before final submission for marking.

3.1 Project Plan

The Project Plan will document the organisational structure of the project team. It must include details of the project schedule (and associated scheduling methods), in addition to documenting management roles and responsibilities. The schedule will also contain resource allocation for all tasks. This will enable tracking of individual and team process, and awareness of major checkpoints and milestones.

3.2 Software Quality Assurance Plan

The SQAP documents all standards, procedures and conventions to ensure effective planning, management and development of all software products. The intended result of this is a quality product.

3.3 Software Requirements Specification

The SRS is intended for both the development team and the client. It must comprehensively specify the requirements of the product, and thus provide a definitive basis for the formulation of the software design.

3.4 Software Design Document

The SDD must show how the software is to be structured. It will describe all components of the system, their interactions, and how each component contributes to satisfying the requirements specified in the SRS.

3.5 User Task Analysis

The UTA describes the users of the future system and outlines the major tasks the Client hopes to accomplish with the product.

3.6 Software Test Plan

The TP will describe what tests are to be performed when testing the software system, as well as the methods used to create test cases. The TP must include sufficient test cases to ensure that the software adheres to the SRS.

3.7 Risk Management Plan

The RMP will show the analysis of the risks that affect the project and propose solutions to minimise the effect of these risks. It will form part of the Project Plan.

3.8 Software Configuration Management Plan

The configuration management plan will describe the methods for version control and general configuration management. It will form part of the Project Plan.

3.9 Project Progress Report

Regular reviews of the progress of the project will be performed during the team meetings. The results of these progress reviews will form the basis of the Project Progress Report.

3.10 User Manual

The user documentation must provide users with comprehensive information regarding usage of the RoboCup software, as well as data requirements, program options and limitations.

3.11 Additional Documents

Additional documentation, such as 'How To' guides, reports and research documents may also be included if deemed useful.

4 Standards, Practices, Conventions and Metrics

4.1 Meeting Procedures

4.1.1 Meeting Attendance

All personnel shall make every attempt to be present at the team meetings. Attendance at the weekly team meeting is mandatory. Team members who cannot attend or will be late to the meeting should notify the chairperson at the earliest practical time. Absenteeism shall be noted in the minutes. Consistent failure to attend team meetings will be investigated by the Team Leader - who will attempt to assist in dealing with the cause.

4.1.2 Calling a Meeting

The appointed chairperson is responsible for calling the meeting. The chairperson will be responsible for organising an email to the expected attendees with the agenda attached. The agenda is also to be placed in the CVS. The publication of the agenda is expected at least 24 hours before the meeting.

4.1.3 Meeting Procedure

The meeting will proceed according to the agenda. The Chairperson is responsible for keeping the discussion relevant and maintaining scheduled finish times as much as is possible. Minutes will be kept for the meeting, and published within 48 hours.

4.1.4 Meeting Agenda Format

The agenda is to be produced using LaTeX; the file should be named according to the following standard:

agenda_$< YY_MM_DD >$.tex

Agenda files are to contain the following:

Header comments in LaTeX file:

FILE:	$< FileName >$
REQ. FILES:	$< RequiredFileNames >$
REQ. PACKAGES:	$< RequiredPackageName >$
DESCRIPTION:	Group Meeting Agenda
PROJECT:	RoboCup 2004
LANGUAGE:	LaTeX2e
AUTHOR:	$< Author >$
CREATED:	$< DateCreated >$

The following settings will be used for LaTeX:

Document class:	[a4paper,10pt]article
Package:	fancyhdr

The sections of the Agenda will be (These may vary slightly according to the Chairpersons requirements):

- Meeting Title
- Date
- Start Time
- Finish Time
- Location
- Purpose
- Minutes of Previous meeting (if applicable)
- Agenda Items

4.1.5 Meeting Minutes Format

The Meeting Minutes are also to be produced using LaTeX; the file should be named according to the following standard:

$< MeetingType >$_Min_$< YY_MM_DD >$.tex

$< MeetingType >$ will be blank for general team meeting.

The Meeting Minutes will contain the following:

Header comments in LaTeX file:

FILE:	$< FileName >$
REQ. FILES:	$< RequiredFileNames >$
REQ. PACKAGES:	$< RequiredPackageName >$
DESCRIPTION:	Meeting Minutes
PROJECT:	RoboCup 2004
LANGUAGE:	LaTeX2e
AUTHOR:	$< Author >$
CREATED:	$< DateCreated >$

The following settings will be used for LaTeX:

Document class:	[a4paper,10pt]article
Package:	fancyhdr

The sections of the Meeting Minutes will be (These may vary slightly according to the Chairpersons requirements):

- Meeting Title
- Date
- Start Time
- Finish Time
- Location
- Purpose
- Attendance
- Items
- Actions with Dates due
- Next meeting Date

4.2 Documentation Standards

All formal documents will be prepared using LaTeX and are to follow the Standard Document Template (found in the `/templates` directory). This template requires the `RoboCup.sty` and `fancyhdr.sty` packages, which can also be found in the templates directory.

All LaTeX source must adhere to these simple guidelines:

- 'Sections' must be preceded by three blank lines.
- 'Subsections' must be preceded by a blank line.
- 'Subsubsections' must be preceded by a blank line.
- The formatting of the cover page, headers/footers, and the table of contents should be based upon the robocup.sty template.
- LaTeX source should be formatted for easy readability in the TeXnicCenter IDE.
- Where external documents are referenced, a bibliography is to be included.
- Code samples in documents must be presented in `typewriter` font.

4.3 Document Naming Standards

All files and directory names shall only consist of the characters
`[A-Z, a-z, 0-9, - *dash*, *dot*, _ *underscore*, *space*, #]`

4.4 Coding Standards

Software for the project must be written according to both the RoboCup Project CS and Swinburne University CS. Where the standards are insufficient, software will be written in a clear and consistent style.

Team members will lay-out code according to the RoboCup Team Template. Templates showing standard banner comments and revision histories can be found on the CVS repository under the `\templates` directory.

4.5 Comment Standards

Comments must be written to comply with the Swinburne University CS. All comments are to be written in Javadoc-style header files. Documentation in source files will mainly consist of inline comments, or comments which add to code readability.

Templates showing standard banner comments and revision histories can be found on the CVS repository under the `\templates` directory. File and class banner comments are to use the Doxygen extensions to Javadoc syntax.

4.6 Testing Standards

Test standards will be described in the Test Plan.

4.7 Timelog Standard

A timelog will be added to the CVS repository for each student, and updated on a weekly basis. It will contain the following information as a minimum:

- Student name
- Student ID
- Swinburne Week Number (as heading)
- Subtotal of hours for week

For each timelog entry, the following information will be recorded:

- Date that the work was performed
- Number of hours spent on the work
- Short description (one sentence) of the work performed

To avoid confusion pertaining to the recording of part-hours, a decimal fraction of an hour is to be used in lieu of a 'pseudo-decimal' representation of minutes. For example, 0.5 hours will be used to signify a half-hour, vis-a-vis 0.3 hours. Timelogs are to be made in quarter-hour (i.e. 0.25) increments; work periods falling in-between these increments may be rounded up to the nearest quarter-hour.

The name of the timelog file will use the following convention:
"TimeLog-< $StudentName$ >.txt"

4.8 Communications Standards

Communications standards have been defined to promote efficient and accurate passing of information between the relevant stakeholders. They are as follows:

4.8.1 E-Mail Standards

Methods: All team members are required to check their e-mail at least once on every weekday. All e-mail relating to the project will have the word ”RoboCup” in the subject line.

An attempt will be made to limit the number of e-mails sent on any one subject to three per day.

Team members should send e-mails only to the relevant parties. This will ensure that multiple recipient e-mails are not glossed over by any team member assuming that it is not relevant to them.

Logging: A sub-directory should be created for storage of all RoboCup-related e-mails.

4.8.2 Out-of-office standards

Methods: The white-board in EN202 will be used to record ‘Out-of-office’ status for each of the students. If a student will not be attending university on any given weekday, this is to be recorded on the white-board. Absence of a student’s name will be assumed to indicate availability at the university.

4.8.3 Meeting Agenda Communication

Methods: The meeting agenda, which will be available at least one day before each meeting, is to be printed out and read by all meeting participants.

Logging: Agendas and meeting minutes will be stored on CVS.

4.8.4 Written Communication Standards

The language standard to be used in all written documents is ‘UK-ise’ (e.g. ‘summarise’, as opposed to ‘summarize’). Furthermore, UK spelling of words — such as ‘colour’ — is the preferred style for all RoboCup documents. There may be exceptions to this rule, such as the use of ‘program’ in favour of the rarely-used ‘programme’, especially in relation to computer programs.

4.9 Metrics and Measures

The metrics and measures used to track and measure RoboCup software development will include:

1. Metric: Size of code
   - Measure: Lines of Code
2. Metric: Coupling and Cohesion
   - Measure: Number of classes
   - Measure: Number of functions
3. Metric: Maintainability
   - Measure: Number of global variables

- Measure: Number of member functions
- Measure: Number of member variables
- Measure: Number of namespaces
- Measure: Number of files

4. Metric: Development Efficiency
   - Measure: Development time
   - Measure: Number of builds
   - Measure: Test successes

4.10 Compliance Monitoring

Metrics will be recorded fortnightly using GCCXML and the python ElementTree XML package, and with a custom add-in for Microsoft Visual Studio .NET. Metrics are to be stored into CSV files. Comparisons of metrics from previous builds will be done using Microsoft Excel. This will allow problem areas, such as inappropriate use of global variables or slow growth, to be identified and addressed immediately.

5 Review

Code for all major deliverables is to be reviewed at least fortnightly. Documents are to undergo a final review prior to release. Reviews must be placed on the agenda for the week's meeting, and confirmation of the review is to be included in the minutes.

The review process is outlined in the Work Flow Diagram in Figure 1 below.

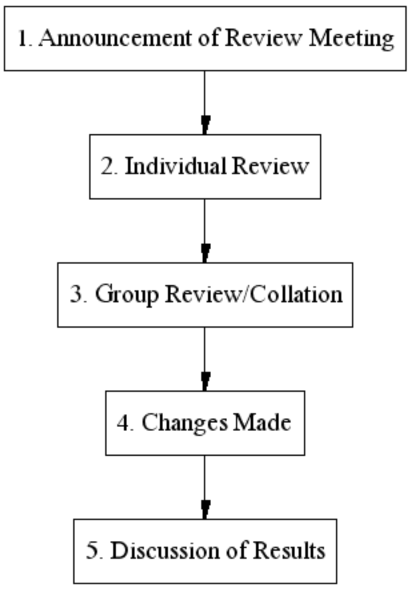

Figure 1: Steps to the Review Process

5.1 Review Procedure

This process is as follows:

5.1.1 Before the Review Meeting

- Appropriate QA officer (code or document) announces review team members, time and location at weekly team meeting.
- If the item is a document, the Team Leader will ensure that the document is located in the /docs/reviews folder within the CVS repository.
- The Team Leader will notify all reviewers that the document/code is ready for review.
- Each review team member acquires hard copy of code/document and performs individual review prior to group review. Corrections and suggestions are to be indicated on the hard copy via pen.

5.1.2 During the Review

- Team members meet at the designated time, bringing with them their marked-up hard copy.
- A formal read-through of the document will take place, with all items arising from each member of the review board being discussed.
- Each reviewer gives a recommendation for the document:
  - Pass;
  - Pass with modifications;
  - Fail (document will require a re-review).
- Reviewers convey any recommendations back to the team.

5.1.3 After the Review

- All hard copies of the review reports will be stored with the Team Leader.
- The author(s) of the code/document makes the required changes, as agreed during the meeting.
- The document is re-reviewed, if required. Otherwise, the result of the changes are discussed during the next weekly team meeting.

6 Testing

Testing will be carried out in a systematic way. Exact details of testing will be included in the Test Plan (TP). Problems with test results will be investigated at the weekly team meetings. Failure to produce sufficient or appropriate test code will be detected in the review process.

Prototype code does not require a formal test suite. Prototype code which is reviewed with the intention of it becoming 'stable' will require an appropriate test suite.

6.1 Test Procedure

Test cases for unit testing will be created before the unit code is written. This will ensure that the requirements for the unit are explicitly addressed before the coding commences. Thus, unit testing will form an integral part of the development of the code. Immediately after building the code, the unit tests will be run to confirm that the code possesses the desired functionality. Non-conformance will be identified during the development stage, rather than an explicitly defined 'Testing stage'.

Integration tests must be conducted regularly by product team-leaders. Higher-level testing will be organised by the team as a whole. The full testing procedure, including regression testing, will be automated to run daily. Test results will be recorded to CSV files.

6.2 Evaluation of Test Reports

Test reports will be formed via the analysis of the raw test data in CSV format, in Microsoft Excel. This will enable comparison between the number of errors versus time, and type of errors versus time, and will give an indication of the effectiveness of the testing methods, and the module/system itself.

The inclusions that are to be made in a test report will be defined during the review of the test cases for each module.

Test reports will be subject to team review during the weekly team meetings, to determine any required changes to software design, implementation, or testing method.

6.3 Client Validation

As soon as the system is deemed capable by the project team, the Client will be given a walk-through to verify that the system satisfies the requirements specified in the SRS.

7 Problem reporting and corrective action

7.1 Defect Reporting

All bug reports are to be entered into the bug tracker database on mercury. Team members must ensure that they include in the report all necessary steps to reproduce the bug.

7.2 Defect Correction

Defects in the database will be reviewed at the weekly team meetings. Assignment of bug correction tasks will be conducted by the sub-team responsible for the defective product.

7.3 Conflict Resolution Process

Refer to 2004 RoboCup Project Plan (section 3.6)

8 Tools, Techniques and Methods

8.1 Software Development Life Cycle

The project will be scheduled in accordance with a hybrid SDLC model. This will incorporate techniques from Throwaway Prototyping, Iterative, and the Waterfall Model. Due to the large number of loosely-coupled products required for the project, most development will be carried out incrementally. Please refer to the Project Plan for more information.

8.2 Choice of Language

Embedded software running on the robots will be written in C or Assembly as required. The application layer code will be in C++. Rapid-prototyping and configuration scripts will be written using Lua.

8.3 Tools for Coding

All embedded code will be written using the ImageCraft IDE. All application-layer code will be written using the Microsoft Visual Studio .NET IDE. Monitoring of Visual Studio .NET will be conducted through a custom add-in, recording events within the IDE.

9 Code Control

All data (source code, documentation, configuration files etc.) must reside in the group's CVS repository on mercury. Temporary files created when compiling documents or source code must not be added to the repository. The directory structure for the main repository may be seen in table 1, which outlines the directories and their contents.

10 Records Collection, Maintenance and Retention

All records, including agendas, minutes and review records must be retained for the duration of the project. As much material as practicable must be kept in the CVS repository. Material from the 2003 RoboCup team will be checked into the repository to allow access to older material. A backup from CVS will be retained after the repository has been locked by the SEP staff, and then any further material will be added to aid the ongoing success of the RoboCup project.

11 Training

Refer to the PP for details of how training will be conducted. Problems relating to the quality or volume of training will be brought up at the weekly team meeting.

12 Risk Management

Risks are to be identified and described in the Project Plan. They are to be assessed and ranked according to each risk's likelihood, impact and exposure. A risk management plan will be employed to reduce the threat posed by these factors. It must outline the procedures to be applied in the event of 'risk realisation', and is to be designed with project quality in mind.

Directory	Description
`- + admin`	Linux administration scripts
`- + code`	*No code to be stored directly in /code*
`  + - embedded`	stable embedded hardware code
`  + - stable`	stable releases (post-review with tests)
`    + + common`	Shared Source
`    + + imgproc`	Image Processing Files
`    + + tactics`	Tactics source
`  + - prototypes`	software prototypes (pre-review code)
`    + + common`	Shared Source
`    + + embedded`	prototype embedded code
`    + + imgproc`	Image Processing Files
`    + + tactics`	Tactics source
`    + + marketing`	Code for marketing demonstrations
`    + + yourPrototypeHere`	storage at user's discretion
`  + - vspaths`	customised visual studio setup files
`- + datasheets`	data-sheets for electrical components
`- + docs`	*No docs to be stored directly in /docs*
`  + - ADD`	Architectural Design Doc
`  + - agendas`	meeting agendas
`  + - CS`	Coding Standard
` + - Inventory`	Inventory of all Robocup equipment
`  + - minutes`	meeting minutes
`  + - misc`	Miscellaneous information
`  + - PP`	Project Plan
`  + - Reviews`	Documents pending review
`  + - SDD`	Software Design Doc
`  + - SQAP`	Software Quality Assurance Plan
`  + - SRS`	Software Requirements Specification
`  + - STP`	Software Test Plan
`  + - thesis`	Robotics final report
`  + - timesheets`	timesheets / worklogs
`  + - TP`	Technology Prototype Report
`  + - UID`	User Interface Design Report
`  + - User Manual`	User Manual/Documentation
`  + - UTA`	User Task Analysis
`  + - Weekly_status_reports`	Weekly reports of team progress
`- + drawings`	Robocup drawings (typically AutoCAD)
`- + templates`	All templates (code, header, documents etc.)
`- + test`	Storage for general or CVS testing
`  + - yourTestHere`	storage at user's discretion

Table 1: Directory Structure

Appendices

A Required "After Release" Changes

This section outlines the changes we would make to this document with the benefit of hindsight. The body of the SQAP remains the same as that specified after the last team review on 05/07/2004.

The required changes are as follows:

Calling a meeting: We initially specified that the meeting agendas were to be emailed to all team members prior to the meeting. In practice, since the meetings were scheduled regularly, the agendas were placed on CVS and the responsibility left to the individual team members to read the agenda prior to the meeting.

Comment standards We should have specified that MFC-generated code does not require comments.

Timesheets We specified that the Swinburne week number should be included in the timesheets, but this does not account for holidays. A standard for week numbers which took into account holidays should have been presented.

Reviews It was not practical to conduct reviews each fortnight. When code was deemed to be at a stage whereby a review was beneficial, then it was reviewed. Marked-up copies were not stored with the team leader. They were of much better use to the person who had to make the code changes!

Evaluation of Test Reports If failure occurred during unit tests, they were generally fixed straight away, or within 1 or 2 days (whatever time was required to solve the problem). Explicit analysis of test failures vs. time was not deemed necessary in the case of unit tests. Build error logs, on the other hand were examined in a similar manner to that defined in the "Evaluation of Test Reports" section.

Defect Reporting Bug-tracking software was installed, but not used. Persistent bugs were outlined during team meetings and informal discussions. The meeting minutes served as a practical means to keep track of on-going bugs.

Security Although no problems were reported, team members should ensure that no user name or password should be stored in a file within the repository.

File Naming Conventions The presence of spaces and unexpected periods (.) within file names can cause problems for some software. Future RoboCup teams are recommended to adopt a stricter policy with regards to the naming of files. In particular Visual Studio solution and project files (.sln and .vcproj), which cannot be used from the command line if there are spaces in the filenames.