Applying Broadcasting/Multicasting/Secured Communication to agentMom in Multi-Agent Systems

Software Quality Assurance Plan

Version 1.1

This document conforms to IEEE Std 730.1-1995 Software Quality Assurance Plan.
This document is submitted in partial fulfillment of the requirements for the degree MSE.

Chairoj Mekprasertvit
CIS 895 – MSE Project
Kansas State University
Spring 2003

Table of Contents

1 Purpose

The purpose of this document is to specify how the software quality assurance plan will be handled in the software development life-cycle of the project "Applying Broadcasting/Multicasting/Secured Communication to agentMom in Multi-Agent Systems". The intended use of this to software project is to enhance the pre-exist agentMom (agentMom 1.2) to be capable of providing multicasting conversation and basic message encryption for security purpose. This document is based on the IEEE standard for Software Quality Assurance Plan, IEEE Std 730.1-1995. This document is intended to be used in partial fulfillment of the requirements for the Master of Software Engineering Project's Portfolio. Furthermore, this document will be reviewed and evaluated by the major professor and the supervisory committee.

2 Reference

- Project Overview version 1.0, Kansas State University, March 2003, (http://www.cis.ksu.edu/~cme6556/project_overview_1.0.pdf)
- Project Plan version 1.0, Kansas State University, March 2003, (http://www.cis.ksu.edu/~cme6556/project_plan_1.0.pdf)
- Software Requirements Specification, Kansas State University, March 2003, (http://www.cis.ksu.edu/~cme6556/software_requirements_specification_1.0.pdf)
- IEEE Guide for Software Quality Assurance Planning
- IEEE Standard for Software Quality Assurance Planning, IEEE Std 730.1-1995
- Software Project Management: A Unified Framework
- AgentMom User's Manual, Air Force Institute of Technology, July 2000
- Multiagent Systems Engineering. *The International Journal of Software Engineering and Knowledge Engineering*

3. Management

3.1 Organization

Supervisory Committee consisted of:

Dr. Scott A. DeLoach
Dr. David Gustafson
Dr. William Hankley

Major Professor:

Dr. Scott A. DeLoach

Developer:

Chairoj Mekprasertvit

Formal Technical Inspector consisted of

Madhukar Kumar
Acharaporn Pattaravanichanon

3.2 Responsibilities

3.2.1 Supervisory Committee

Primary responsibilities include reviewing each milestone deliverable at the requirements, architecture, and implementation phases. After reviewing, each committee member should provide feedback and suggestions to the software developer.

3.2.2 Major Professor

In addition to the responsibilities as one of the committee member, the major professor will supervise and evaluate all artifacts submitted by developer. Reviews and walkthroughs of related materials will be conducted on weekly basis.

3.2.3 Software Developer

Since the project is being developed individually, developer is responsible for ensuring quality of the project. The software developer is also responsible for producing the required artifacts for MSE projects.

3.2.4 Formal Technical Inspectors

The major responsibility of inspectors is to provide a formal report on their inspection result from architecture design artifact produced by developer.

Furthermore, the following tasks will be conducted in order to ensure quality assurance:

1. Driving Requirements: The developer should ensure that the software requirement specification in the vision document clearly states the functionality of software and unambiguously declares the requirements that must be satisfied. In addition, descriptions of the scope should clearly outline what the software will allow and not allow.

2. Design: The developer and major professor will conduct reviews and analyses of the construction of the software. Strengths and weaknesses of various design techniques will be discussed and scrutinized.

3. Implementation: Informal code reviews will be conducted by the developer on a regular basis to ensure consistency with the design and the detection of any error. Also, JavaDoc will be produced for purpose of maintainability and future work.

4. Testing: Developer will conduct tests as presented in the Software Test Plan to ensure the requirement satisfaction and reliability of the software.

4. Documentation

The following documentation will be generated and updated throughout the duration of software life cycles:

Phase I:

1.) Vision Document - provides detailed description of the entire project, goals of the software, constraints and requirements for the software to satisfy.
2.) Project Plan - illustrates the major milestones and provides a rough timeline for the project and estimation on the size and effort of the project.
3.) Software Quality Assurance Plan – provides plan for software quality assurance

Phase II:

1.) Formal Requirement Specification – UML/OCL methodology will be used to produce this document.
2.) Test Plan - provides description of test cases during testing
3.) Architecture Design – Object Model and Use Cases will be produced.
4.) Formal Technical Inspection - two MSE students will participate in formal technical inspection, and developer will also provide an inspection checklist.

Phase III:

1.) User Manual - instructions on how to use software

2.) Final source code - actual implemented documented source code
3.) Assessment Evaluation - assessment of reliability and performance of software
4.) Project Evaluation - review of the entire project

5. Standards, Practices, Conventions and metrics

5.1 Standards

- Documents – MSE portfolio requirements, CIS Dept., Kansas State University
- Coding – Java 1.4.0 (commenting will follow JavaDoc standards)
- Testing – IEEE Standard for Software Test Documentation

5.2 Metrics

- SLOC – source lines of code will be primarily used for measuring the size of the software
- COCOMO I – cost estimation will be calculated based on COCOMO I model.

6. Reviews and audits

Two formal MSE students will perform a formal technical inspection on the architecture design document and provide a formal report. Also, each committee member will review the produced documentation and make comments and suggestions during each presentation. Each milestone must be approved by each committee member to proceed to the next milestone. Each milestone is indicated by the presentation of each phase. There are three presentation described as follow:

Presentation I at the end of phase I includes project overview, software requirements, project plan, SQA plan and prototype demonstration.

Presentation II at the end of phase II includes formal requirement specification, architecture design, test plan and architecture prototype demonstration.

Presentation III at the end of phase III includes component design, assessment evaluation, project evaluation, result from formal technical inspection and completed software demonstration.

7. Problem reporting

If any problems are encountered throughout the duration of the project, the software developer can report and discuss the problems with the major professor. If any conflicts or problems are discovered by one of the committee members during a presentation, the developer will then correct the errors.

8. Tools, Techniques and Methodologies

For determining whether the software requirements were satisfied, a software test plan will be written during phase II. This plan will provide an overview of the methodologies, timetables, and resources for testing the software. Testing will commence in three primary ways:

8.1 Unit Testing

Individual classes will be tested to ensure reliability and functionality within a unit-level. Furthermore, testing module will be created before the tested code to ensure that

the code is testable. Junit 3.8 will be the tool to perform testing. Unit testing will be performed before alpha and beta release.

8.2 Integration Testing

Several classes will be tested together to ensure sufficient execution and compliance with the requirements after integration. Integration testing will be performed before beta release.

8.3 System Testing

The whole system shall be used for system testing to ensure all requirements is satisfied, and reliability will be included in the testing to measure successful rate of message delivery. System testing will be performed before beta release.

The following tools are used in creating, testing and debugging software

- Java 1.4.2 will be the language used for coding the software.
- USE 2.0 will be used for modeling the formal specifications, using UML/OCL methodology
- Rational Rose will primarily be used for producing object models.
- Eclipse IDE 2.1 will be used for coding the software package.
- This software package will be tested under Microsoft Windows XP/2000, Linux Debian and Unix Solaris 9.

9. Media control

All the required documentation generated throughout the course of the project is available at the software developer's personal website (http://www.cis.ksu.edu/~cme6556).

Upon project completion, a CD containing the entire project document, prototype, and final product is created.

10. Document and Software Version Control

Each document version or software version is incremented by 0.1 after it is approved by major professor. Also, version is move to the next digit after it is approved by all committee members. For example, version 1.2 will be changed to version 2.0 after all committee approval or version 1.2 to 1.3 after major professor approval.
Alpha version is released after the software is passed all unit tests.
Beta version is released after the software is passed all unit tests, integration tests and system tests.
Final version is released after the software is approved by major professor.

11. Training

CIS 740 Software Engineering
CIS 748 Software Management
CIS 771 Software Specifications
CIS 890 Agent-Oriented Software Engineering