Software Quality Assurance Plan
Mastergoal Machine Learning Environment
Version 0.1.0

Submitted in partial fulfillment of the requirements of the degree of MSE

Alejandro Alliana
CIS895 – MSE Project
Index:

1. **Purpose** .............................................................................................................. 1
2. **Reference Documents** ....................................................................................... 1
3. **Management** ........................................................................................................ 1
   3.1. Organization ....................................................................................................... 1
   3.2. Tasks .................................................................................................................. 2
   3.3. Responsibilities .................................................................................................. 2
4. **Documentation** .................................................................................................... 3
   4.1. Purpose .............................................................................................................. 3
   4.2. Minimum Documentation Requirements ......................................................... 3
   4.3. Other ................................................................................................................ 4
5. **Standards, practices, conventions and metrics** .................................................. 4
   5.1. Purpose .............................................................................................................. 4
   5.2. Documentation Standards .................................................................................. 4
   5.3. Logic structure standards ................................................................................... 4
   5.4. Coding Standards .............................................................................................. 4
   5.5. Commentary standards ...................................................................................... 4
   5.6. Testing standards and practices ........................................................................ 5
   5.7. Selected software quality assurance product and process metric .................... 5
6. **Reviews and Audits** ............................................................................................ 5
7. **Test** ..................................................................................................................... 6
8. **Problem reporting and corrective action** ............................................................. 6
9. **Tools, Techniques and methodologies** ................................................................. 6
10. **Code Control** ..................................................................................................... 7
11. **Media Control** .................................................................................................. 7
12. **Supplier Control** ............................................................................................... 7
13. **Records Collection, maintenance and retention** ................................................. 7
14. **Training** ............................................................................................................. 7
15. **Risk management** ............................................................................................. 8
Changes:

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Revision</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>07/22/2007</td>
<td>Alejandro Alliana</td>
<td>0.0.2</td>
<td>Initial version</td>
</tr>
<tr>
<td>08/09/2007</td>
<td>Alejandro Alliana</td>
<td>0.0.4</td>
<td>Complete document revision</td>
</tr>
<tr>
<td>08/16/2007</td>
<td>Alejandro Alliana</td>
<td>0.1.0</td>
<td>Presentation 1 Baseline</td>
</tr>
</tbody>
</table>
1. Purpose

The purpose of this document is to provide a SQAP for the “Mastergoal Learning Environment” project. The Mastergoal Learning Environment (MLE) is an application to be used to create experiments and learn strategies for the game of Mastergoal using Artificial Intelligence techniques.

In detail, the purposes of this document are:

- To establish a model and define rules to check if the software process is following all the standards.
- To ensure that all the evaluation plans, test cases, and development are well followed and documented for the further review.
- To define what and how the reviews and audits are performed. Also, it defines what actions must be performed after the review and audit is done.
- To ensure that a modern software development approach and process are followed.
- To ensure that all functionality mentioned in the vision document is being implemented in the software developed.
- Scope: The Software Quality Assurance Plan for the MLE will describe all the phases of a modern Software development approach. That is, it covers the inception, elaboration, construction, and transition phases as described by Walker's Royce modern software approach.

2. Reference Documents


3. Management

3.1. Organization

This section describes the organizational structure of the MLE project.
### 3.2. Tasks

The tasks are defined in detail in the Project Plan [Al07-2], an overview of the tasks is given here to make the document self-contained. This SQAP covers all phases of the modern lifecycle as defined by Royce [Ro98]. This particular process consists in three phases as shown in

Again, a detailed description of tasks and artifacts is given in [Al07-2]. Special artifacts strongly related to quality assurance are:

- Prototypes and demonstrations created and performed in the inception and elaboration phases.
- Vision document.
- Formal technical inspections
- Architecture design.

![Figure 1. Phases of the lifecycle.](image)

### 3.3. Responsibilities

**Developer** The developer is responsible for documentation, implementation, and testing of this project. Three presentations will be given detailing project progress. Bi-weekly meetings with the major professor are required; these meetings are also open to all members of the committee.
The major professor is responsible for overseeing the work done by the developer. He must approve all artifacts of the project and is considered the end-user for the MLE validation. He will also provide guidance and advice to the developer throughout the project’s lifetime.

The supervisory committee is responsible for preparing for and attending each of the three presentations. The committee will provide comments on the artifacts presented during the presentation and feedback is expected at these times.

The formal technical inspectors are responsible for completing a formal inspection of the project framework. They will also conduct a preliminary functional walkthrough of the project.

4. Documentation

4.1. Purpose

The purpose of this section is to identify the documentation governing the development, the verification and validation procedures and maintenance procedures of MLE.

4.2. Minimum Documentation Requirements

<table>
<thead>
<tr>
<th>PHASE</th>
<th>DOCUMENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td>• Engineering Notebook</td>
</tr>
<tr>
<td></td>
<td>• Vision Document</td>
</tr>
<tr>
<td></td>
<td>• Project Plan</td>
</tr>
<tr>
<td></td>
<td>• Prototype</td>
</tr>
<tr>
<td></td>
<td>• Software quality assurance plan.</td>
</tr>
<tr>
<td>Architecture</td>
<td>• Vision Document</td>
</tr>
<tr>
<td></td>
<td>• Project Plan</td>
</tr>
<tr>
<td></td>
<td>• Formal Requirement Specification</td>
</tr>
<tr>
<td></td>
<td>• Architecture design.</td>
</tr>
<tr>
<td></td>
<td>• Test Plan</td>
</tr>
<tr>
<td></td>
<td>• Formal technical specification</td>
</tr>
<tr>
<td></td>
<td>• Prototype</td>
</tr>
<tr>
<td>Implementation</td>
<td>• User manual</td>
</tr>
<tr>
<td></td>
<td>• Component design</td>
</tr>
<tr>
<td></td>
<td>• Source Code</td>
</tr>
<tr>
<td></td>
<td>• Assessment evaluation</td>
</tr>
<tr>
<td></td>
<td>• Project evaluation</td>
</tr>
<tr>
<td></td>
<td>• Formal technical inspection letters</td>
</tr>
</tbody>
</table>
4.3. **Other**

The software development plan is presented in another document.

5. **Standards, practices, conventions and metrics**

5.1. **Purpose**

The compliance to the standards defined in this section will be monitored and assured before each of the scheduled presentations by the developer and the committee.

5.2. **Documentation Standards**

The requirements should be documented assigning a unique number to each requirement. Each use case should be documented using Use Case diagrams in UML and describing it in a table in the following format.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>{use case number and name.}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td></td>
</tr>
<tr>
<td>Goal</td>
<td></td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
<tr>
<td>Scenario</td>
<td></td>
</tr>
<tr>
<td>Post conditions</td>
<td></td>
</tr>
<tr>
<td>Exceptions</td>
<td></td>
</tr>
<tr>
<td>Open issues</td>
<td></td>
</tr>
</tbody>
</table>

The design artifacts should be documented in rigorous model based notations that allow round trip engineering, such as UML, ER diagrams, etc.

All documents must be available online for downloading.

5.3. **Logic structure standards**

Not identified at this moment.

5.4. **Coding Standards**

The coding standards of the C/C++ language will be followed.

5.5. **Commentary standards**

Source code should be documented following a style similar to the comments standard suggested by Sun in the *Code Conventions for the Java Programming Language* available at [http://java.sun.com/docs/codeconv/](http://java.sun.com/docs/codeconv/)
5.6. **Testing standards and practices**

Testing will be done using automatic testing suites and following the *Continuous Integration* approach. The details about how testing will be done are explained in the *Test plan*.

There are going to be three types of testing that will be carried out during the process:

- **Unit testing**: This is the testing that will be carried out at the class or function level.
- **Integration testing**: This testing will be done to test the interaction of different components together.

Testing is done in all the phases of development.

5.7. **Selected software quality assurance product and process metric**

A number of different metrics are being considered as the cost estimating metrics. These metrics include COCOMO, COCOMO II, and Use case Points.

Some other metrics that should be collected are:

- Number of Open and Closed defects found, with aging and trending over a specified time frame
- Number of Peer Reviews (Planned vs. Actual)
- SQ effort and funds expended (Planned vs. Actual)
- Number of Software Quality Assessments (Planned vs. Actual)
- Number of Software Quality Assessment Findings or non compliances (Open vs. Closed)
- Number of Software Quality Assessment Observations
- Number of Risks identified as a result of an Software Quality Assessment

6. **Reviews and Audits**

The developer will review the documents, source code and models periodically. The advisor might review the artifacts when needed.

Artifacts will be reviewed by the committee after the conclusion of each phase and before the presentation.

Formal inspections will be carried in the architecture and implementation phases of the project.
7. Test

To verify that the product complies with the requirements, the following types of testing are required:

- Component testing
- Interface testing.
- System testing
- Performance testing.
- Documentation testing.

Modern testing techniques such as automated, regression testing as well as continuous integration will be done. The complete testing procedures will be specified in the Testing Plan.

8. Problem reporting and corrective action

Each problem found in each development phase must be identified, categorized and registered in a defect database. This data measurement will provide information not only for assessment but also for process improvement. The solution to the problem must also be registered in the database to use as reference when the problem reappears.

If the error is one which requires changes throughout one or more items then as many corrections as possible should be made.

It is the responsibility of the developer to ensure that all code committed to the central repository compiles and runs correctly.

All problems or suggestions made by the committee or major professor must be documented and studied. The problem must be solved by the next presentation.

9. Tools, Techniques and methodologies

Requirements and other Architectural models will be documented using use case diagrams in the Rational Rose modeling tool. This tool was selected because it allows round trip engineering. Textual notations for requirements will be maintained using the Microsoft Word and adobe PDF format.

Microsoft Visual Studio 2005 will be used as an Integrated Development Environment for the project.

A subversion repository will be used as a code control repository. Tortoise SVN will be used as a client tool to access the repository.
Unit testing, software inspections, requirements tracing, requirements and design verification techniques should be used in the process.

10. **Code Control**

All source code and artifacts will be maintained in a subversion\(^1\) version control system. The *Tortoise SVN*\(^2\) tool will be used to access the repository.

Source code must be tagged as a release after each major milestone (presentation) and at minor milestones considered relevant.

11. **Media Control**

Documents will be available on the internet after each presentation and the approval of the committee.

The head (trunk) version of the artifacts will be stored in the developer computer protected by password.

12. **Supplier Control**

This section is not applicable to this project at the moment.

13. **Records Collection, maintenance and retention**

All documents and source code should be kept in softcopy in the backups of the system.

After the completion of the project all the artifacts will be provided to the major professor and will follow the department rules for master’s portfolio.

14. **Training**

The following table represents the technology knowledge requirements for each of the tasks of the software development process:

---

### Development Tasks

<table>
<thead>
<tr>
<th></th>
<th>Development Tasks</th>
<th>Management Tasks</th>
<th>Testing Tasks</th>
<th>Environment Tasks</th>
<th>Design Tasks</th>
<th>Requirements Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>UML</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>C++</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual Studio IDE</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVN</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Project</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Word</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Operating System Administration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Cpp test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>

### 15. Risk management

**Schedule control**: The developer will control and continuously update the schedule, analyzing delays in milestones and checking the impact of these delays on the overall project schedule.

**Early risks resolution**: Risks will be solved early by software demonstrations.

**Continuous integration** methodology will be used to try to detect any inconsistencies in the product as soon as possible.

**Automated testing**: Automated testing should be used to avoid regression bugs.

**Round trip engineering**: This technique helps maintaining the architectural models of the system synchronized with the source code without spending a lot of time.