Postmortem Evaluation

Mastergoal Machine Learning Environment

Version 0.3

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

Alejandro Alliana

CIS895 – MSE Project
Change Log:

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Revision</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/22/2008</td>
<td>Alejandro Alliana</td>
<td>0.1</td>
<td>Initial version</td>
</tr>
<tr>
<td>04/23/2008</td>
<td>Alejandro Alliana</td>
<td>0.2</td>
<td>Complete revision</td>
</tr>
<tr>
<td>05/01/2008</td>
<td>Alejandro Alliana</td>
<td>0.3</td>
<td></td>
</tr>
</tbody>
</table>
Index

1. Introduction .................................................................................................................. 2
2. Problems found ................................................................................................................. 2
3. Metrics ............................................................................................................................ 2
   3.1. Lines of Code ............................................................................................................. 2
   3.2. Time ........................................................................................................................... 5
4. Projects estimates ............................................................................................................. 7
5. Lessons learned ............................................................................................................... 8
6. Future work ....................................................................................................................... 9
7. References ....................................................................................................................... 9
1. Introduction

This document evaluates the experience of the development of the MMLE project. An evaluation of the tools, process, techniques employed as well as the mistakes made is presented so that lessons are documented and learned. In the last section future direction of the project is presented.

2. Problems found

The main problem encountered during the process was my inexperience in the C++ language and environment. Another problem was my inexperience in the use of design patterns. Sometimes patterns were overused.

The project includes some advanced topics, like mixing managed and unmanaged C++ code. Some bugs in the IDE (Microsoft Visual Studio 2005 version 2.0.50727) were found that required lots of time of research to find a solution. i.e. at some point the entry point of the .NET project had to be changed in the linker\footnote{Entry point was set to "?mainCRTStartupStrArray@ @$FYMHP$01APAAVString@System@@Z"}, there is not a lot of documentation about this bug that was fixed only for the visual studio 2008.

3. Metrics

3.1. Lines of Code

Metrics where computed weekly using the CCCC (C and C++ Code Counter). CCCC extracts many metrics from the source code, I only used the lines of code (LOC), number of modules (NOM) and comment lines (COM) metrics as shown in Figure 1 for the whole project. The total number of LOC is 12456 (Apr-30-2008)
The complete results of the metrics for each sub-project are available online at:

7. UI Metrics: [http://people.cis.ksu.edu/~aalliana/895/metrics-mgWinFormApp/cccc.html](http://people.cis.ksu.edu/~aalliana/895/metrics-mgWinFormApp/cccc.html)

The big jump in the LOC between the 5th and 12th of March is because the TinyXML sources were added to the Mastergoal sub-project at that time (Figure 2). The TinyXML sources have 1239 lines of code.

Figure 2, Figure 3, Figure 4 and Figure 5 show charts for individual metrics for each sub-project.

![mastergoal project](image-url)  
Figure 2. Mastergoal project metrics
Figure 3. MGAI project metrics

Figure 4. MMLE project metrics

Figure 5. UI Project metrics
3.2. Time

The project took 31981 minutes, which is 533 hours, at 40 hours a week that is 13.3 weeks or 3 months. Figure 6 shows the time worked per phase. The implementation phase is by far the phase where most of the work was done as it can be seen in Figure 6.

The tasks where categorized by phase, workflow and task. Figure 7 and Figure 8 show the time spent at each task and at each workflow. The tasks that are more important are coding, testing and design.

Figure 9 shows time spent for each task at each phase. Design is the most important task that was performed at two different phases (elaboration and implementation). Coding was performed in all phases but mainly in the implementation phase.

![Figure 6. Time spent per phase](image-url)
Figure 7. Time spent per task

Figure 8. Time spent for each workflow.
4. Projects estimates

The estimated size of the project was 2.79 KLOC using function points and 7.5 KLOC estimate when using COCOMO. Both estimates were short, since the project is almost 11 KLOC (subtracting 1 KLOC of the TinyXML sources).

The difference to the estimate using function points can be attributed to a) The lack of experience of using function points b) Some of the user interfaces were more complex than previously thought and c) A big part of the project is the user interface which contains automatically generated code, which is very large and is not consistent with the LOC conversion rates of the Function point estimates.

The size of 7.5 KLOC used in COCOMO was arbitrary, but was also short. Again reasons (a) through (c) of the previous paragraph apply. Furthermore, in the estimates the testing projects were not considered. The sum of LOC of the thee test projects is of about 1 KLOC as can be seen in Figure 102

The project estimates of about 11 months was about right (the project took 10 months), but I consider this merely a coincidence, since the size of the project was miss-

---

2 This figure does not include the TinyXML LOC.
estimated and the delivered artifacts should have taken less time than the overhead estimated by the COCOMO method. Furthermore, considering a person working full time on the project, it would take only 3 months of time. The inexperience in the C++ made the progress of the project very slow at times.

![Size of each Project (LOC)](image)

Figure 10. LOC per sub-project.

5. Lessons learned

There are many aspects of learning during the project.

**Implementation:** The C++ language was obviously the most important area of learning, but also the implementation of design patterns in this language (and the relation to other languages) was important.

The use of many new tools like cppunit and visual leak detector was very important.

**Design:** Again, the implementation of design patterns was difficult at times. Many patterns were applicable in some situations and I decided to try them all to see which one was better. Also some language issues make difficult the implementation of the design patterns.

I did have some problems trying to find a reasonable UML tool that worked fine with C++ code. Because of the memory management scheme of the C++ language, the UML diagrams are more complex than in a language with garbage collector. I.E. The difference between the aggregation and composition relation is much more evident. A tool that allows full roundtrip engineering and worked fine was not found.

**Estimates:** Estimates are still my weak point. I assume that experience will help me to make better estimates in the future. The experience gained using the Function Points and COCOMO methods was important.
Measurement: Measurement proved to be important but it would be interesting to compare the results of the metrics with other projects now that I have more experience with the language and environment.

Testing: Finding regression bugs with the testing tools was satisfactory, but the downside was the amount of code and time spent creating these test classes. Also, now that I have more experience with the tools it would be interesting to use them in other projects. As a deficit in testing a more systematic approach of documenting bugs should be followed.

Process: An iterative approach was followed. As a deficit in the Inception and Elaboration more prototyping should have been done. The project was useful to gain experience in the preparation of several artifacts. This experience should be useful to tailor the process in future projects to try to keep the overhead as minimal as possible.

6. Future work

Training evaluation functions is very time consuming. One possible improvement is to change the architecture of the system to perform parallel distributed computation. This project did not follow this approach but the separation of the different sub-projects was done so that different sub-projects could be reused.

Refactoring of some aspects in the different sub-projects:

- Mastergoal library:
  - Refactoring of the Board and Rules classes. Some of the operations of the Rules class are more appropriate to the Board class.
  - Refactoring of the Position instantiation: Position should have a factory method that creates positions as singletons.

- Mastergoal AI library:
  - Use of a node interface.
  - Implementation of more Search Algorithms.
  - Cleaning up the Machine Agent interface.

- MMLE:
  - Implementation of more learning techniques (Reinforcement Learning)
  - Implementation of more strategies (non linear evaluation functions).

7. References