Project Plan

Mastergoal Machine Learning Environment

Version 0.1.4

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>06/22/2007</td>
<td>Alejandro Alliana</td>
<td>0.1</td>
<td>Initial version</td>
</tr>
<tr>
<td>06/22/2007</td>
<td>Alejandro Alliana</td>
<td>0.2</td>
<td>Initial version release</td>
</tr>
<tr>
<td>08/10/2007</td>
<td>Alejandro Alliana</td>
<td>0.3</td>
<td>Included COCOMO II estimates.</td>
</tr>
<tr>
<td>08/11/2007</td>
<td>Alejandro Alliana</td>
<td>0.4</td>
<td>Sent to Dr. Gustafson for review before presentation.</td>
</tr>
<tr>
<td>08/16/2007</td>
<td>Alejandro Alliana</td>
<td>0.5</td>
<td>Minor formatting modifications.</td>
</tr>
<tr>
<td>08/16/2007</td>
<td>Alejandro Alliana</td>
<td>1.0</td>
<td>Presentation I Baseline</td>
</tr>
<tr>
<td>08/26/2007</td>
<td>Alejandro Alliana</td>
<td>1.1</td>
<td>Committee recommendations: Adding function points</td>
</tr>
<tr>
<td>08/26/2007</td>
<td>Alejandro Alliana</td>
<td>1.2</td>
<td>Final version including committee comments after first presentation.</td>
</tr>
<tr>
<td>11/03/2007</td>
<td>Alejandro Alliana</td>
<td>1.3</td>
<td>Updated FP estimate and added Gantt diagrams</td>
</tr>
<tr>
<td>11/04/2007</td>
<td>Alejandro Alliana</td>
<td>1.4</td>
<td>Version to be included in the Baseline for Presentation II</td>
</tr>
</tbody>
</table>
Index

Project Plan

1. Project Plan

1.1. Inception phase

1.1.1. Engineering Notebook

1.1.2. Vision Document

1.1.3. Project Plan

1.1.4. Prototype demonstration

1.1.5. Software Quality Assurance Plan

1.2. Elaboration phase (Architecture)

1.2.1. Action Items

1.2.2. Vision Document

1.2.3. Project Plan

1.2.4. Formal Requirements Specification

1.2.5. Architecture Design

1.2.6. Test Plan

1.2.7. Formal Technical Inspection

1.2.8. Executable Architecture Prototype

1.3. Implementation phase

1.3.1. Action recording and documentation

1.3.2. User Manual

1.3.3. Component Design

1.3.4. Project Coding

1.3.5. Assessment

1.3.6. Project Evaluation

1.3.7. Formal technical Inspection

2. Cost Estimate

2.1. COCOMO

2.2. COCOMO II

2.2.1. Applications composition model

2.2.2. Early design model

2.2.3. Post Architecture model

2.3. Function Points

2.3.1. Determining function points by type:

2.3.1.1. External Inputs (EI)

2.3.1.2. External Outputs (EO):

2.3.1.3. External Inquires (EQ):

2.3.1.4. Internal Logical Files (ILF):

2.3.1.5. External Interface Files:

2.3.2. General System Characteristics:

2.3.3. Calculating the adjusted function point:

2.3.4. Cocomo II:

3. Project Timeline

4. Architecture Elaboration Plan

4.1. Activities

4.2. Requirements formalization

4.3. Formal technical inspections
References:


COCOMO II Model Definition Manual, Dr. Barry Boehm, Chris Abts, Brad Clark, Sunita Devnani-Chulani

Function Point Analysis Training Course, David Longstreet. www.softwaremetrics.com
Project Plan

The project will be divided in three phases: Inception, Elaboration and Implementation according to the MSE Portfolio requirements\(^1\)

1.1. Inception phase

1.1.1. Engineering Notebook

The engineering notebook records the activities, effort and if possible the productivity of these activities. The measurement of the effort spent in the different activities is part of step 0 in the Personal Software Process suggested by Watts Humphries. Because of the importance of errors, the errors as well as the time spent in identifying and removing the error are also recorded.

1.1.2. Vision Document

The vision document has two objectives:

- The first goal is to provide a complete overview of the software system being developed, its main characteristics, quality requirements, etc.
- The second goal is to provide a description of the driving requirements of the project. These requirements are described using visual modeling languages (UML) as well as text descriptions.

1.1.3. Project Plan

This document describes the process used to create the product. It also includes a timeline of the project and a cost estimate for the production process. Gantt charts are used to present a project schedule.

The project plan also includes an architecture elaboration plan (AEP). The AEP defines the activities and actions that must be accomplished prior to the Architecture Presentation. The AEP also includes the names of two MSE students that will do a formal technical inspection of the project.

1.1.4. Prototype demonstration

The prototype has the goal of demonstrate a solution for the risky elements of the requirements. This approach follows the confront risks early principle presented by Walker Royce\(^2\).

1.1.5. Software Quality Assurance Plan

This document describes the documentation, standards and conventions used to develop the product. It also describes the tools used throughout in all the phases of the process. The

---

\(^1\) [http://mse.cis.ksu.edu/oncampus/mse-portfolio.htm](http://mse.cis.ksu.edu/oncampus/mse-portfolio.htm)

\(^2\) Software Project Management, a unified framework. Walker Royce, Addison Wesley 1998
document describes the test procedures as well as the defect recording and removal processes.

The document presents the quality metrics used in the process and the product reliability expected.

1.2. **Elaboration phase (Architecture)**

The elaboration phase goal is to present an architecture design and demonstration for the project. This phase also provides revisions to the artifacts created in the inception phase. Two formal reviews of the architecture are performed by other MSE students.

1.2.1. **Action Items**

The effort spent on the revision of all the items specified in the previous phase must be recorded and documented.

1.2.2. **Vision Document**

The changes suggested by the committee in the first presentation as well as the changes due to the natural evolution of the vision document must be made to the vision document.

1.2.3. **Project Plan**

The changes suggested by the committee in the first presentation as well as the changes due to the natural evolution of the vision document must be made to the project plan.

1.2.4. **Formal Requirements Specification**

A subset of the project will be specified using a formal specification language such as alloy or OCL.

1.2.5. **Architecture Design**

The complete architectural design will be documented using appropriate UML diagrams such as class and object diagrams, sequence/collaboration diagrams, activity diagrams. Each component in the architecture will be documented at the interface level. Reuse of commercial or pre-existing components will be documented.

1.2.6. **Test Plan**

The test plan will present the required test that demonstrates that the product satisfies the requirements presented in the vision document and the formal requirements specification. The plan will include evaluation criteria for the critical use cases, test logs and test documents to be produced, as well as the test methodologies, procedures and test tools to be used.

1.2.7. **Formal Technical Inspection**
Two other MSE students will perform a formal technical inspection of one of the technical artifacts. A formal checklist to be used by the inspectors will be provided and the inspectors will provide a report of the result their inspection.

### 1.2.8. Executable Architecture Prototype

An executable architecture prototype will be built. The prototype will address all critical requirements identified in the vision document and expose the top technical risks.

### 1.3. Implementation phase

#### 1.3.1. Action recording and documentation

The effort spent on the revision of all the items specified in the previous phase must be recorded and documented.

#### 1.3.2. User Manual

A user manual will be created. Sections will include an overview and explanations of common usage, screens, user commands, error messages, and data formats.

#### 1.3.3. Component Design

The internal design of each component will be documented. The documentation required will be consistent with the complexity of the individual components. The use of model-based diagrams such as class diagrams, sequence/collaboration diagrams, and state chart/activity diagrams will be considered.

#### 1.3.4. Project Coding

Every component will be coded and kept documented. Source code must be kept synchronized with the design. The source code will be provided at the end of the process.

#### 1.3.5. Assessment

Unit testing will be performed on the source classes. The CppUnit testing tool has been considered for this task. Integration testing will be performed on each iteration after unit testing. Error rate diagrams and other measures will be kept to measure the rate of errors during testing. At the end of the process an Assessment Evaluation document will be provided. This document will include information about the methodologies used and results obtained in the process.

#### 1.3.6. Project Evaluation

After the conclusion of the project, a Project Evaluation document will be presented. This document must include the evaluation of the process, methodologies used. The document must also include a review of the accuracy of the estimators, and the usefulness of the reviews. The resulting product will be contrasted against the initial overview and the quality presented in the vision document.

---

3 http://cppunit.sourceforge.net/cppunit-wiki
1.3.7. Formal technical Inspection
The student must participate in two other MSE projects as an inspector.

2. Cost Estimate
The document will also provide a detailed estimate on the size, cost and effort required for the project. At the conclusion of the project, the estimation effort will be critiqued.

In this section the effort is estimated using the traditional COCOMO model and the COCOMO II model. For COCOMO II, the three sub models are estimated at this point, and will be updated during the process.

2.1. COCOMO
The COCOMO model has three modes: Organic, Semidetached, and embedded. For this project the organic mode is appropriate. The cost estimation relationships for the organic mode are:

\[
\text{Effort} = 3.2 \times \text{EAF (Size)}^{1.05}
\]
\[
\text{Time} = 2.5 \times (\text{Effort})^{0.38}
\]

Where:
Effort is the number of staff months
EAF is the product of 15 effort adjustment factors.
Size is the number of delivered source instructions in KLOC.

<table>
<thead>
<tr>
<th>Id</th>
<th>Effort Adjustment Factor</th>
<th>Parameter Range</th>
<th>Potential Impact</th>
<th>Value Selected</th>
<th>Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Required reliability</td>
<td>0.75 - 1.40</td>
<td>1.87</td>
<td>1.00</td>
<td>Nominal - The application is reliability is not critical</td>
</tr>
<tr>
<td>DATA</td>
<td>Database size</td>
<td>0.94 - 1.16</td>
<td>1.23</td>
<td>1.00</td>
<td>Nominal - Database access to store games</td>
</tr>
<tr>
<td>CPLX</td>
<td>Product complexity</td>
<td>0.70 – 1.65</td>
<td>2.36</td>
<td>1.15</td>
<td>High – Product contains reinforcement learning algorithms.</td>
</tr>
<tr>
<td>TIME</td>
<td>Execution time constraint</td>
<td>1.00 – 1.66</td>
<td>1.66</td>
<td>1.11</td>
<td>High – Experiments must not take too long, since the application is already computationally intensive.</td>
</tr>
<tr>
<td>STOR</td>
<td>Main Storage Constraint</td>
<td>1.00 – 1.56</td>
<td>1.56</td>
<td>1.00</td>
<td>Nominal</td>
</tr>
<tr>
<td>VIRT</td>
<td>Virtual machine volatility</td>
<td>0.87 – 1.30</td>
<td>1.49</td>
<td>1.00</td>
<td>Nominal</td>
</tr>
<tr>
<td>TURN</td>
<td>Computer turnaround time</td>
<td>0.87 – 1.15</td>
<td>1.32</td>
<td>1.00</td>
<td>Nominal</td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst capability</td>
<td>1.46 – 0.71</td>
<td>2.06</td>
<td>0.86</td>
<td>High. Developer has adequate experience.</td>
</tr>
<tr>
<td>AEXP</td>
<td>Applications experience</td>
<td>1.29 – 0.82</td>
<td>1.57</td>
<td>1.10</td>
<td>Low. Some of the components are new to the developer</td>
</tr>
<tr>
<td>PCAP</td>
<td>Programmer capability</td>
<td>1.42 – 0.70</td>
<td>2.03</td>
<td>1.00</td>
<td>Nominal</td>
</tr>
<tr>
<td>VEXP</td>
<td>Virtual machine experience</td>
<td>1.21 – 0.90</td>
<td>1.34</td>
<td>1.00</td>
<td>Nominal. Developer has adequate experience with OS systems and tools.</td>
</tr>
<tr>
<td>LEXP</td>
<td>Language experience</td>
<td>1.14 – 0.95</td>
<td>1.20</td>
<td>1.07</td>
<td>Low. Developer is new to the C++ language.</td>
</tr>
<tr>
<td>MODP</td>
<td>Use of modern practices</td>
<td>1.24 – 0.82</td>
<td>1.51</td>
<td>0.91</td>
<td>High. The process will follow modern practices.</td>
</tr>
</tbody>
</table>
The estimated size measured in LOC is 7.5 KLOC. I think this estimate is quite pessimistic and can be considered as a higher bound. Other MSE projects estimated their projects between 2 and 3 KLOC but the resulting products contained 5 to 6 KLOC.

\[
\text{Effort} = 3.2 \times 1.18 \times 7.5 \times 1.05 = 31.32 \text{ staff months}
\]

\[
\text{Time} = 2.5 \times (31.32)^{0.38} = 9.25 \text{ months}
\]

### 2.2. COCOMO II

COCOMO II defines three models for cost estimation:
- Applications composition model
- Early design model
- Post-Architecture model.

The three models will be computed at this point, so they can be verified during the process. I think this might be useful as a way of understanding the models.

#### 2.2.1. Applications composition model

The applications composition model corresponds to exploratory work typically done during prototyping efforts.

The procedure for creating the applications composition model is:

- Assess Object-Counts: estimate the number of screens, reports, and 3GL components that will comprise this application. Assume the standard definitions of these objects in your ICASE environment.
  - Main screen.
  - Game exploration screen
  - Export strategy screen
  - Experiments status report.
  - Explore game screen.
  - Game component.
  - Search component
  - Learning component

- Classify each object instance into simple, medium and difficult complexity levels depending on values of characteristic dimensions.
Object complexity classification:

<table>
<thead>
<tr>
<th>Object</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main screen</td>
<td>Medium</td>
</tr>
<tr>
<td>Game exploration screen</td>
<td>Medium</td>
</tr>
<tr>
<td>Export strategy screen</td>
<td>Simple</td>
</tr>
<tr>
<td>Experiments status report</td>
<td>Simple</td>
</tr>
<tr>
<td>Explore game screen</td>
<td>Medium</td>
</tr>
<tr>
<td>Game component</td>
<td>Simple</td>
</tr>
<tr>
<td>Search component</td>
<td>Medium</td>
</tr>
<tr>
<td>Learning component</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 2. Object complexity classification.

Classification scheme for screens and reports:

<table>
<thead>
<tr>
<th>Number of Views contained</th>
<th>For Screens</th>
<th>For Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># and source of data tables</td>
<td># and source of data tables</td>
</tr>
<tr>
<td></td>
<td>Total &lt; 4 (&lt; 2 srvr</td>
<td>Total &lt; 8 (2/3 srvr</td>
</tr>
<tr>
<td></td>
<td>&lt; 3 clnt)</td>
<td>3-5 clnt)</td>
</tr>
<tr>
<td>&lt; 3</td>
<td>simple</td>
<td>simple</td>
</tr>
<tr>
<td>3 - 7</td>
<td>simple</td>
<td>medium</td>
</tr>
<tr>
<td>&gt; 8</td>
<td>medium</td>
<td>difficult</td>
</tr>
</tbody>
</table>

Table 3. Classification scheme for screens and reports

- Weight the number in each cell using the following scheme. The weights reflect the relative effort required to implement an instance of that complexity level.

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Complexity-Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple</td>
</tr>
<tr>
<td>Screen</td>
<td>1</td>
</tr>
<tr>
<td>Report</td>
<td>2</td>
</tr>
<tr>
<td>3GL Component</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Weights for each object complexity type.

- Determine Object-Points: add all the weighted object instances to get one number, the Object-Point count.

<table>
<thead>
<tr>
<th>Object</th>
<th>Classification</th>
<th>Object Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main screen</td>
<td>Medium</td>
<td>2</td>
</tr>
<tr>
<td>Game exploration</td>
<td>Medium</td>
<td>2</td>
</tr>
</tbody>
</table>
• Estimate percentage of reuse you expect to be achieved in this project. Compute the New Object Points to be developed.

\[
\text{NOP} = (\text{Object Points}) \times \frac{(100-r)}{100} = 36
\]

• Determine a productivity rate, \( PROD = \frac{\text{NOP}}{\text{person-month}} \), from the following scheme

<table>
<thead>
<tr>
<th>Developers’ experience and capability</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICASE maturity and capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD</td>
<td>4</td>
<td>7</td>
<td>13</td>
<td>25</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 6. Productivity ranges.

Considering the experience of the developer in the programming language and that the project will be developed part time because the developer has other subject I considered that the productivity low is adequate for the project.

• Compute the estimated person-months:

\[
PM = \frac{\text{NOP}}{\text{PROD}}
\]

Equation 1. Person months formula.

\[
PM = \frac{39}{7} = 5.57 \text{ Person months}
\]

2.2.2. Early design model

In the early design model, the effort can be computed using the following formula:

\[
\text{Effort} = 2.45 \ E_{\text{Arch}} \ (\text{Size})^p
\]

Where:
Effort = number of staff-moths  
\( E_{\text{Arch}} \) = is the product of seven early design effort adjustment factors  
Size = number of function points or KLOC  
P = process exponent.

<table>
<thead>
<tr>
<th>Early Design Cost Driver</th>
<th>Counterpart Combined Post-Architecture Cost Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCPX</td>
<td>RELY, DATA, CPLX, DOCU</td>
</tr>
<tr>
<td>RUSE</td>
<td>RUSE</td>
</tr>
<tr>
<td>PDIF</td>
<td>TIME, STOR, PVOL</td>
</tr>
<tr>
<td>PERS</td>
<td>ACAP, PCAP, PCON</td>
</tr>
<tr>
<td>PREX</td>
<td>AEXP, PEXP, LTEX</td>
</tr>
<tr>
<td>FCIL</td>
<td>TOOL, SITE</td>
</tr>
<tr>
<td>SCED</td>
<td>SCED</td>
</tr>
</tbody>
</table>

Table 7. Early design cost driver and post architecture cost drivers relation.

Using this table and assigning values to each of the seventeen post architecture drivers we come up with the values of the following table.

<table>
<thead>
<tr>
<th>Early design cost driver</th>
<th>Abbreviation</th>
<th>Value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Reliability and Complexity</td>
<td>RCPX</td>
<td>Nominal – 4.21</td>
</tr>
<tr>
<td>Required Reuse</td>
<td>RUSE</td>
<td>Very High – 1.49</td>
</tr>
<tr>
<td>Platform difficulty</td>
<td>PDIF</td>
<td>Nominal – 3.11</td>
</tr>
<tr>
<td>Personnel capability</td>
<td>PERS</td>
<td>High – 2.70</td>
</tr>
<tr>
<td>Personnel Experience</td>
<td>PREX</td>
<td>Nominal – 3.07</td>
</tr>
<tr>
<td>Facilities</td>
<td>FCIL</td>
<td>Nominal – 1.69</td>
</tr>
<tr>
<td>Schedule</td>
<td>SCED</td>
<td>High – 1.10</td>
</tr>
</tbody>
</table>

Total: 300.61

Table 8. Cost drivers values for the Early design model.

<table>
<thead>
<tr>
<th>Scale Factors</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC</td>
<td>thoroughly unprecedented</td>
<td>largely unprecedented</td>
<td>somewhat unprecedented</td>
<td>generally familiar</td>
<td>largely familiar</td>
<td>thorough familiar</td>
</tr>
<tr>
<td>FLEX</td>
<td>rigorous</td>
<td>occasional relaxation</td>
<td>some relaxation</td>
<td>general conformity</td>
<td>some conformity</td>
<td>general goals</td>
</tr>
<tr>
<td>RESL(7)</td>
<td>little (20%)</td>
<td>some (40%)</td>
<td>often (60%)</td>
<td>generally (75%)</td>
<td>mostly (90%)</td>
<td>full (100%)</td>
</tr>
<tr>
<td>TEAM</td>
<td>very difficult interactions</td>
<td>some difficult interactions</td>
<td>basically cooperative interactions</td>
<td>largely cooperative</td>
<td>highly cooperative</td>
<td>seamless interactions</td>
</tr>
<tr>
<td>PMAT</td>
<td>Weighted average of &quot;Yes&quot; answers to CMM Maturity Questionnaire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9. Scale factors for COCOMO II.

<table>
<thead>
<tr>
<th>Scale Factor</th>
<th>Abbreviation</th>
<th>Value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precedentedness</td>
<td>PREC</td>
<td>Nominal – 3</td>
</tr>
</tbody>
</table>
B = 1.01 + 0.01 * Sum (3 + 4 + 3 + 0 + 5)
B = 1.01 + 0.01 * 15
B = 1.01 + 0.15
B = 1.16

Using the same estimate of 7.5 KLOC used for COCOMO I.

Effort = 2.45 (300.61) (7.5)\(^{1.16}\)
Effort = 736.49 (10.35)
Effort = 7622.67 staff months

This result does not make any sense. I could not find any documentation that helps clarify this result so I’m going to take note of this as a bug and try to correct the problem for next iteration.

### 2.2.3. Post Architecture model

The post architecture model is pretty similar to the COCOMO I model, but with some changes in the adjustment factors. These are the adjustment factors of COCOMO II and the selected values.

<table>
<thead>
<tr>
<th>Id</th>
<th>Effort Adjustment Factor</th>
<th>Parameter Range</th>
<th>Value Selected</th>
<th>Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Required reliability</td>
<td>0.75 - 1.39</td>
<td>1.00</td>
<td>Nominal - The application is reliability is not critical</td>
</tr>
<tr>
<td>DATA</td>
<td>Database size</td>
<td>0.93 - 1.19</td>
<td>1.00</td>
<td>Nominal - Database access to store games</td>
</tr>
<tr>
<td>CPLX</td>
<td>Product complexity</td>
<td>0.70 – 1.66</td>
<td>1.15</td>
<td>High – Product contains reinforcement learning algorithms.</td>
</tr>
<tr>
<td>RUSE</td>
<td>Required reusability</td>
<td>0.91 – 1.49</td>
<td>1.49</td>
<td>Very High</td>
</tr>
<tr>
<td>DOCU</td>
<td>Documentation</td>
<td>0.95 – 1.13</td>
<td>1.06</td>
<td>High</td>
</tr>
<tr>
<td>TIME</td>
<td>Execution time constraint</td>
<td>1.00 – 1.67</td>
<td>1.11</td>
<td>High – Experiments must not take too long, since the application is already computationally intensive.</td>
</tr>
<tr>
<td>STOR</td>
<td>Main Storage Constraint</td>
<td>1.00 – 1.57</td>
<td>1.00</td>
<td>Nominal</td>
</tr>
<tr>
<td>PVOL</td>
<td>Platform volatility</td>
<td>0.87 – 1.30</td>
<td>1.00</td>
<td>Nominal</td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst capability</td>
<td>1.50 – 0.67</td>
<td>0.86</td>
<td>High. Developer has adequate experience.</td>
</tr>
<tr>
<td>AEXP</td>
<td>Applications experience</td>
<td>1.22 – 0.81</td>
<td>1.10</td>
<td>Low. Some of the components are new to the developer</td>
</tr>
<tr>
<td>PCAP</td>
<td>Programmer capability</td>
<td>1.37 – 0.74</td>
<td>1.00</td>
<td>Nominal</td>
</tr>
<tr>
<td>PCON</td>
<td>Personnel continuity</td>
<td>1.24 – 0.84</td>
<td>0.84</td>
<td>Extremely High – All components will</td>
</tr>
</tbody>
</table>
Table 11. COCOMO II cost drivers assignment.

<table>
<thead>
<tr>
<th>Driver</th>
<th>PEXP</th>
<th>LEXP</th>
<th>SITE</th>
<th>TOOL</th>
<th>SCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform experience</td>
<td>1.25 – 0.81</td>
<td>1.00</td>
<td>Nominal. Developer has adequate experience with OS systems and tools.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language experience</td>
<td>1.22 – 0.84</td>
<td>1.07</td>
<td>Low. Developer is new to the C++ language.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-site development</td>
<td>1.25 – 0.78</td>
<td>0.78</td>
<td>Extra High- Fully collocated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use of software tools</td>
<td>1.24 – 0.72</td>
<td>0.91</td>
<td>High.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required development schedule</td>
<td>1.29 – 1.00</td>
<td>1.10</td>
<td>Low. Project is on a constrained schedule</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Effort = 2.45 \left( E_{\text{app}} \right) (\text{Size})^P

Where:
- Effort = number of staff-moths
- \( E_{\text{Arch}} = \) is the product of seventeen post architecture effort adjustment factors of the previous table.
- Size = number of function points or KLOC
- P = process exponent, same as the early design model.

\[
\text{Effort} = 2.45 \times (1.34) \times (7.5)^{1.16}
\]

Effort = 33.99 staff months

\[
\text{Time} = 2.5 \times \text{(Effort)}^{0.38}
\]

Time = 9.54 months

As expected, COCOMO and the post-architecture model of COCOMO II have similar values. The results for other COCOMO II models are somewhat different and will be verified during the process.

### 2.3. Function Points

To compute an estimate using function points, the following procedure must be followed:

- Determine function points by types:
  - External inputs
  - External outputs
  - External Inquires
  - Internal Logical files
  - External interface files.
- Determine the complexity-level function counts.
- Apply complexity weights
- Compute the unadjusted function points
- Compute the general system characteristic factor
- Determine the type of function points to be computed
  - Development project function point count
  - Enhancement project function point count
Application function point count
  • Compute the adjusted function point count.

2.3.1. Determining function points by type:

2.3.1.1. External Inputs (EI)

**Experiment Window:** This window adds changes and deletes information of the experiment, so it is considered as three external inputs.

![Prototype of the main experiment window](image)

Number of data elements: 19  
Number of File type references: 1  
Score: Average (4) see Table 12.

Since this window is used to create, modify and delete experiments, the total number of EI function points is $3 \times 6 = 18$
GA-Data Window:

Number of data elements: 13
Number of File type references: 1
Score: Low (3) see Table 12.

Since this window is used to create, modify and delete GA-data, the total number of EI function points is $3 \times 3 = 9$

**Game play/explore Window:** This window is used to play a game against a selected strategy or to explore saved games.
Number of data elements: 16
Number of File type references: 1
Score: Average(4) see Table 12.

Since this window is used to create and modify games, the total number of EI function points is $4 \times 2 = 8$

Total number score for EI is 35

2.3.1.2. **External Outputs (EO):**

**Export Strategy Window:** This window is used to export a strategy to an external file format.
Number of data elements: 5
Number of File type references: 2
Score: Average(4) see Table 12.

Since this window is used to create and modify games, the total number of EI function points is $4 \times 2 = 8$

<table>
<thead>
<tr>
<th>File Types Referenced (FTR)</th>
<th>Data Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>6-19</td>
</tr>
<tr>
<td>Less than 2</td>
<td>Low (4)</td>
</tr>
<tr>
<td>2 or 3</td>
<td>Low (4)</td>
</tr>
<tr>
<td>Greater than 3</td>
<td>Average (3)</td>
</tr>
</tbody>
</table>

Table 13. External output score table.

**GA-Results Window:** This window shows the result of the training using the GA plugin.
Number of data elements: 5
Number of File type references: 4
Score: High(7) see Table 12.

Total score for EO is 15

**2.3.1.3. External Inquires (EQ):**

---

**Experiment Plugin drop down list.**
- Number of Data elements: 1
- Number of FTR: 1
- Score: Low (4), see Table 14

**Terms List**
- Number of Data elements: 1
- Number of FTR: 1
- Score: Low(4), see Table 14

**Search Algorithm**
- Number of Data elements: 1
- Number of FTR: 1
- Score: Low (4), see Table 14

**Strategy drop down list.**
- Number of Data elements: 1
- Number of FTR: 1
- Score: low (4), see Table 14

The total number of EQ function points is 16

---

<table>
<thead>
<tr>
<th>File Types Referenced (FTR)</th>
<th>Data Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-5</td>
</tr>
<tr>
<td>less than 2</td>
<td>Low (3)</td>
</tr>
<tr>
<td>2 or 3</td>
<td>Low (3)</td>
</tr>
<tr>
<td>Greater than 3</td>
<td>Average (4)</td>
</tr>
</tbody>
</table>

Table 14. EQ score table

Total score for EQ is 16

---

**2.3.1.4. Internal Logical Files (ILF):**

---
Table 15. ILF score table.

<table>
<thead>
<tr>
<th>Record Element Types (RET)</th>
<th>Data Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 to 19</td>
</tr>
<tr>
<td>1 RET</td>
<td>Low (7)</td>
</tr>
<tr>
<td>2 to 5 RET</td>
<td>Low (7)</td>
</tr>
<tr>
<td>6 or More RET</td>
<td>Average (10)</td>
</tr>
</tbody>
</table>

**Experiment:**
- Data Elements: ~40
- RET: 2
- Score: High (15)

**Epochs:**
- Data Elements: < 20
- RET: 1
- Score: low (7)

**Population:**
- Data Elements: < 20
- RET: 1
- Score: Low (7)

**Individual:**
- Data Elements: < 20
- RET: 1
- Score: Low (7)

**Game (Moves):**
- Data Elements: ~30
- RET: 2
- Score: Avg (10)

The total number of ILF function points: 46

**2.3.1.5. External Interface Files:**
Not identified at this moment.

**Total unadjusted function points: 112**

**2.3.2. General System Characteristics:**

<table>
<thead>
<tr>
<th>General System Characteristic</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Communications</td>
<td>0</td>
</tr>
<tr>
<td>Distributed Data Processing</td>
<td>0</td>
</tr>
<tr>
<td>Performance</td>
<td>1</td>
</tr>
<tr>
<td>Heavily used configuration</td>
<td>1</td>
</tr>
</tbody>
</table>
Transaction rate 1
On-line data entry 1
End user efficiency 3
Online update 1
Complex processing 3
Reusability 4
Installation ease 0
Operational ease 0
Multiple sites 0
Facilitate change 1

VAF = 0.65 + ∑ Ci / 100
VAF = 0.86

2.3.3. Calculating the adjusted function point:

The development project function point (DFP) calculation

DFP = (UFP + CFP) * VAF

Equation 2. Development function point formula.

Where:
DFP is the development function point count.
UFP is the unadjusted function point count.
CFP is the function points added by supporting applications to convert data and other activities.
VAF: Is the value adjustment factor.

DFP = (112) * 0.86
DFP = 96.32

The total adjusted function points for the project is 54.18

2.3.4. Cocomo II:

To use the adjusted function points in the COCOMO model, the AFP must be converted to LOC. This can be done using the rates of Table 16
Table 16. Function points LOC conversion rates

Using these values, the project estimated size (in KLOC) is 2.79 (96.32 * 29)

Effort = 2.45 (1.34) (2.79)^1.16
Effort = 10.79 staff months

This estimate will be contrasted against the final project values at the postmortem evaluation.

3. Project Timeline

The following diagram presents a timeline of the activities to be done in the implementation phase of the project.
4. Architecture Elaboration Plan

The Architecture Elaboration plan will define the activities and actions that must be accomplished prior to the Architecture Presentation. The plan must include the set of requirements to be formalized and the artifacts that will undergo formal technical inspection.

4.1. Activities

(See section 1.2)

4.2. Requirements formalization

The rules about the moves of the pieces have been formally specified in the OCL language. This artifact has been submitted to two other graduate students for inspection (see formal inspections.

The Architecture has been specified using the UML language and also submitted for inspection.

4.3. Formal technical inspections

Two other MSE students performed a formal technical inspection on the product architecture and the formal specification of the rules for the moves.

Each inspector was provided with a checklist to be used during the inspection process. Each inspector provided a report of the inspection that contains at least the checklist and a cover paper.

The graduate students selected for this task are:

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loai Zomlot</td>
</tr>
<tr>
<td>Kenton Born</td>
</tr>
</tbody>
</table>