Project Evaluation

For agentTool III (Dynamic)

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

Binti Sepaha
CIS 895 – MSE Project
Kansas State University
# Table of Contents

1. Introduction .......................................................................................................................... 3  
2. Problems Encountered ........................................................................................................... 3  
   2.1 Comprehending Project Requirements .................................................................................. 3  
   2.2 Learning GEF (Graphical Editing Framework) ................................................................. 3  
   2.3 Features to Implement ....................................................................................................... 3  
3. Source Lines of Code .............................................................................................................. 4  
4. Project Duration ...................................................................................................................... 4  
5. Lessons Learnt ......................................................................................................................... 5  
6. Future Work ............................................................................................................................. 7
1. Introduction
This document will present a summary of my experiences throughout this MSE Project entitled “agentTool III (Dynamic)“.

2. Problems Encountered
2.1 Comprehending Project Requirements
Initially, understanding the project requirements was tough. Developing the Vision Document took a lot of efforts on my side for understanding the details of the sequence and activity diagrams that are specific to agent-oriented systems. Once the Vision Document was developed, the next problem was visualizing how the tool should be laid out and made very user-friendly. Designing the user interface was one of the most challenging parts of the project. But the toughest to implement was making this tool an eclipse plug-in.

2.2 Learning GEF (Graphical Editing Framework)
In the first phase, I started developing the GUI using Eclipse and Jigloo Plug-in. This was not very difficult but later as the project advanced, I realized that this was not enough for the type of functionality we wanted to provide for these plug-ins. The advanced editing capabilities could only be provided by something like GEF (Graphical Editing Framework). This framework is relatively new and there were not much examples and help online. Understanding the complex GEF had a very steep learning curve. It took almost a month understanding how to build an editor and to combine what parts to get the desired functionality.

2.3 Understanding Plug-in Development
Another tough task was to understand how plug-ins are developed. Eclipse plug-ins need to be built very carefully and require a lot of knowledge on the developer’s part. The learning curve for this was very steep too. But I found helpful manuals and some examples online which made it easier for me to develop the plug-ins.

2.4 Implementation of Some Features
Some features took a lot of time for their implementation. Especially the swim lanes part in the activity diagram editor required lot of efforts. There was a time when I thought that I would not fulfill this requirement, but I kept trying and finally found a solution to it. Another feature was the frame that appears when a new diagram is created. But with constant efforts, I was able to
implement this feature as well. There were other minor things as well that took time but learning all these implementation details was a great experience.

3. Source Lines of Code

The first estimate for SLOC was made during Phase 1 and it was estimated to be 3500 SLOC. This estimate was driven from the previous versions of the tool. I believe this estimate was low because I assumed this estimate when I was using Java with Jigloo. But when I moved to using GEF, the source became larger as it required much other fragments to be combined to build an editor. The next estimate was in Phase 2 and it was estimated to be 5500 SLOC per tool. This estimate was driven from the working Activity Diagram Plug-in which comprised 50% of the actual project. It gave me a better idea of how much code it would take to implement the features in Sequence Diagram Editor. These estimates were accurate as the actual SLOC match these estimates.

The following is a break down of the SLOC required for each plug-in:

- Activity Diagram Editor = **5500**
- Sequence Diagram Editor = **5500**
- Total SLOC = **11000**

4. Project Duration

The following table show the expected vs. actual completion times for each phase of the project.

<table>
<thead>
<tr>
<th>Phase #</th>
<th>Expected Finish Time</th>
<th>Actual Finish Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>October 20, 2004</td>
<td>November 1, 2004</td>
</tr>
<tr>
<td>2</td>
<td>February 25, 2005</td>
<td>March 15, 2005</td>
</tr>
<tr>
<td>3</td>
<td>April 25, 2005</td>
<td>April 25, 2005</td>
</tr>
</tbody>
</table>

The first 2 phases got delayed because of the steep learning curves involved in plug-in development and understanding GEF. The final phase was completed on time.
Time spent in each phase is shown in the following chart:

![Phase Breakdown Chart](image)

**Figure 1. Phase Breakdown**
Following graphs show the breakdown of activities during each phase:

**Figure 2. Phase 1**

**Figure 3. Phase 2**
5. Lessons Learnt
This project was a great learning experience for me. It gave me good exposure to what it is like to work on a large project. I learnt a lot of new things, few of them being Eclipse, Plug-in development, GEF, generating Javadocs, etc. The most important thing I learnt is the time management and how one should effectively schedule his/her work to meet the deadlines. Another valuable lesson I will take away from this project is how thinking through the design will make you think more deeply about what you are building. One should not jump to coding without having a strong design in hand.

6. Future Work
A few features in the Vision Document were marked as future requirements such adding timing constraints to sequence diagrams and implementation of state chart diagrams. Other than these, the features that need to be implemented in future are:

- Adding a message to self
- Adding stop signs to loop and alternative frames. These can be done by making the container policies for the parent objects handle the creation of child like loops and alternative frames can contain tags.