Project Evaluation

For a GMoDS-based Runtime Agent Role Interpreter

Version 1.0

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

Kyle Hill
CIS 895 – MSE Project
Kansas State University

Table of Contents

1 Introduction

This document provides a list of problems encountered, lessons learned, and possible future work for the GMoDS-based Runtime Agent Role Interpreter project. This document also summarizes the number of lines of code in the project as well as various scheduling statistics.

2 Problems Encountered

This section provides a summary of some problems encountered during the project.

2.1 Learning Existing OMACS and GMoDS Frameworks

It took me a lot longer than I expected to learn and understand the OMACS and GMoDS frameworks. I was learning multiagent systems concepts around the same time as I started this project, so I had to devote a lot of time early on to learning basic system concepts and then mapping them code I read within the frameworks. This made for slow development and design tasks at the start of the project.

2.2 Determining Project Requirements

It took quite a while to nail down the exact requirements for this project. I did not get a lot of face-to-face time with my advisor since I am a distance education student. This made for many discussions back and forth over e-mail, adding to the latency when coming up with my initial design and requirements documents

2.3 Time Management

I had many problems with time management on this project. It was hard for me to stay motivated to work on the various tasks required for this project at a consistent rate. Since I am working full time and am a student part time, it was hard for me to juggle the various priorities in my life. I think that this project would have been much easier had I been able to devote myself to it full time.

3 Estimates

This section compares the original estimates in lines of code and total hours of work to the actual totals to provide feedback.

3.1 Source Lines of Code

Phase	SLOC Estimate
Phase 1	5000
Phase 2	4750
Phase 3	5135 (Actual)

Table 1 – Estimates of SLOC by phase

Table 1 shows the total project SLOC estimate by phase. The original estimate of 5000 lines was decreased in the second phase because I did not think an XML parser needed to

be implemented. The actual size grew by about 400 lines in the final phase. The final figure of 5135 SLOC is remarkably close to the original 5000 SLOC estimate.

3.2 Project Duration

Phase	Estimated End Date	Actual End Date
Phase 1	December 8, 2010	December 8, 2010
Phase 2	March 14, 2010	June 23, 2011
Phase 3	May 13, 2010	July 25, 2011

Table 2 – Estimated and actual end dates by phase

Table 2 shows the estimated versus actual end dates for each phase. For the first phase, I knew the end date of the phase because I had already scheduled the presentation by the time I had made the estimate. I greatly underestimated how much work I needed to do in the second and third phases of the project. Due to time pressures from home, work, and other classes, I was not able to devote as much time as I wanted to this project and I had to extend the project out into the summer intersession. I was able to take time off work in the summer and I was able to get a large amount of work done in a short amount of time.

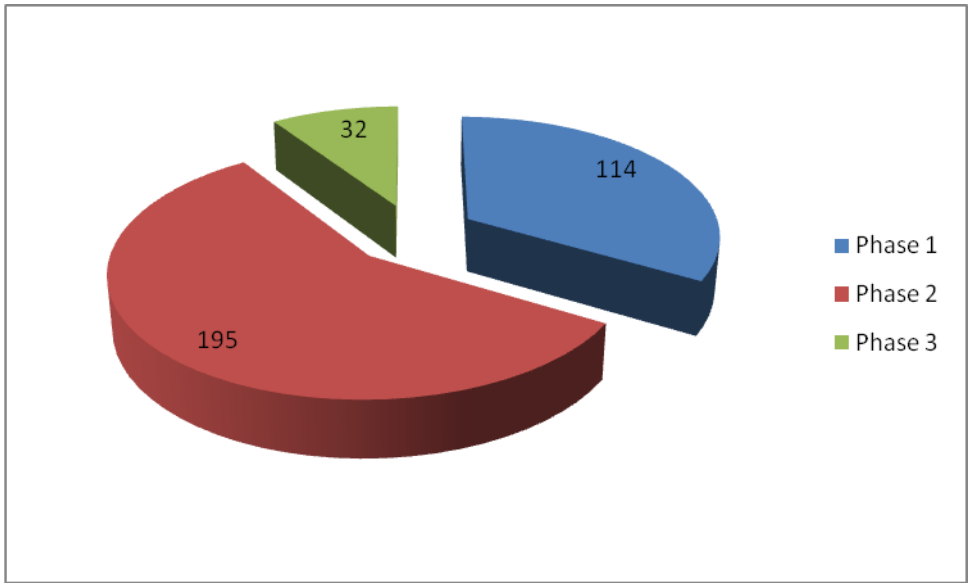

Figure 1 – Percent of time spent in each phase

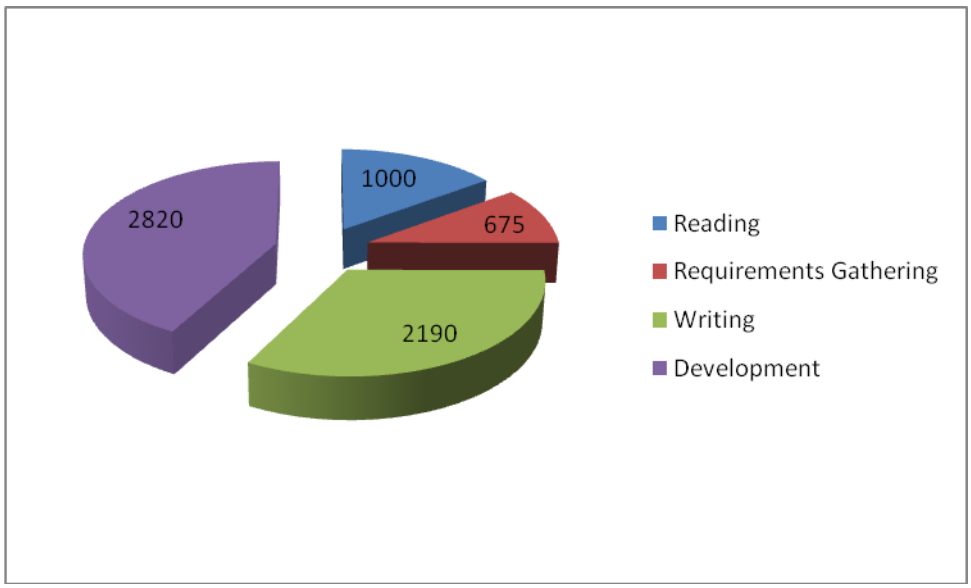

Figure 2 – Percent of time for each task in phase 1

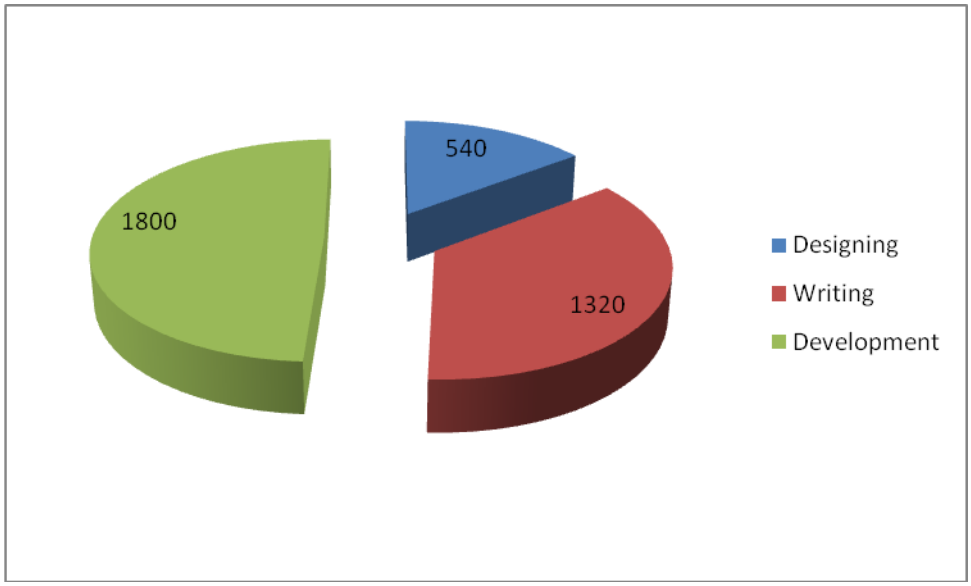

Figure 3 – Percent of time for each task in phase 2

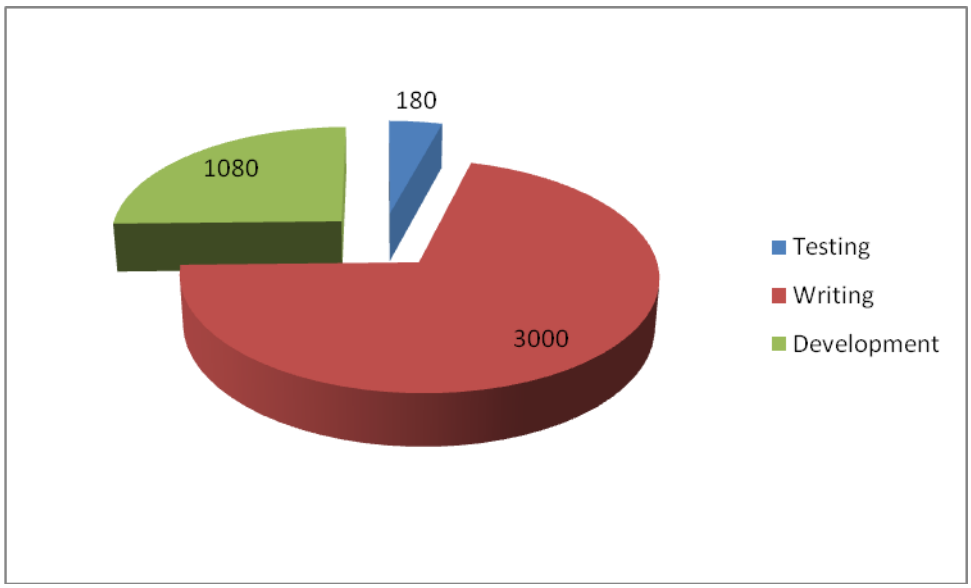

Figure 4 – Percent of time for each task in phase 3

Figures 2 through 4 provide a breakdown, by phase, of the amount of time spent on various tasks throughout the project. At the beginning of the project, I spent a lot more time reading and fleshing out requirements. Near the end of the project, I spent a lot more time testing, finalizing development tasks, and writing documents.

Phase	**Development**	**Writing**	**Designing**	**Testing**	**Requirements**	**Reading**	**Total**
Phase 1	47	36.5	0	0	11.25	17	**111.75**
Phase 2	30	30	9	0	0	0	**69**
Phase 3	18	50	0	3	0	0	**71**
Total	**95**	**116.5**	**9**	**3**	**11.25**	**17**	**251.75**

Table 3 – Time (in hours) for each task by phase

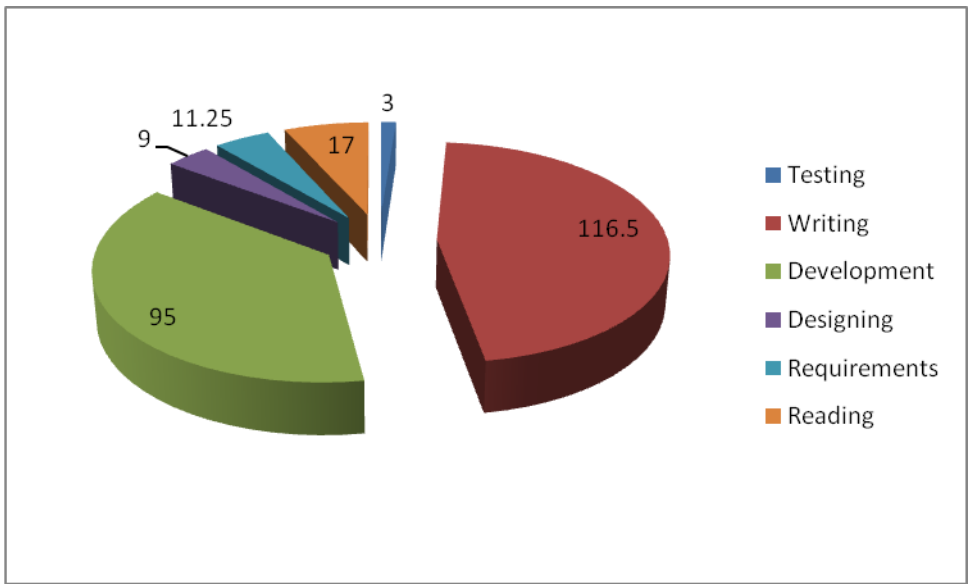

Figure 5 - Percent of time by task

Figure 5 is a graphical representation of the data contained within Table 3. In total, I spent 252 hours working on this project. Nearly half of that time was spent writing documents and another large portion of the time was spent developing the actual software deliverables. I was surprised to see how little time was spent on design, testing, and requirements gathering. Additionally, I was surprised at how much time was spent writing documentation.

4 Lessons Learned

4.1 Computability Theory and Expressiveness

I learned a lot about Computability Theory and Expressiveness concepts, such as Turing Completeness while working on this project. In order to determine the limits of agent behavior specified by Role Level Goal Models, I had to read a lot about these theoretical concepts.

4.2 Java Reflection API

I learned a lot about the runtime Java Reflection API while implementing this project. This powerful interface allows developers to introspectively discover what classes are available to a system, and then determine what methods exist within those classes. This allowed me to programmatically establish goal-capability mappings at runtime without having to know anything about the capabilities agents intended to use beforehand.

4.3 Programming Language Concepts

I learned a lot about various programming language concepts such as the difference between actual and formal parameters and exactly how method overloading works. These concepts were crucial for me to understand when using the Java Reflection API. In

addition, I learned a lot when mapping goal and event parameters to actual method parameters.

5 Future Work

All features and use cases described in the Vision Document were successfully implemented in the project. However, during the course of the project, several areas were identified for possible future enhancements.

5.1 Expand the OMACS Interface Capability

The current OMACS interface capability is rather weak. This capability is intended to provide all the OMACS interactions that a system could need at runtime within a capability that is callable from Role Level Goal Models. Unfortunately, the current capability only allows the creation of hard-coded goals to return Gold to a predefined location. It would be nice to define a more robust and reusable interface for a variety of goal creation and obviation tasks.

5.2 Make a More Complex Agent Architecture Demo

The current agent architecture demo makes use of predefined agent-role-goal assignments. It also does not have any overall organization that controls the assignment of agents to tasks. There is a lot of room for making a more robust system that makes use of an organization to coordinate these activities.

5.3 Remove GMoDS Limitations

A limitation within GMoDS prevents more than one active goal from existing within a "triggers-loop" at one time. I have provided a small patch to GMoDS that makes it possible to actually execute triggers-loops, but it would be nice to make this patch more robust so that it allows more than one active goal at a time. It would also be nice if this patch could be improved so that it is accepted into the official GMoDS framework.