Implementing a remote submission system for the Geometry Friends AI competition

Edgar Santos

Learnings Report

Abstract—I worked to develop a system that automatically detect user submissions form a web site and then compile and execute that code to return the results to the user. Altho the system was not fully functional most of the functionality was implemented.

Index Terms—(Geometry Friends, Artificial Intelligence, AI, Competition, Submission System, LaTeX).

SOFT-SKILLS are NOT SOFTWARE SKILLS

1 Introduction

This report will describe the skills acquired during the development of the Middleware platform for the Geometry Friends automatic testing system, and also some of the techiques used to achieve what was asked. I will start this report by talking about the motivations i had to work on this project, and then move on to the problems faced while working. After that I'll show the solutions implemented to surpass the problems and the finish with with a small conclusion..

2 Motivation

A big part of a Software Engineer's job is the automation of processes to facilitate the life of people, and this was just that. In all my academic career, and outside as well, I used a number of tools that would do repetitive jobs automatically and save countless hours while I was developing, and since I never had any experience in making this kind of software, this served as a great motivator to get started.

3 Problems

There where a lot of problems that needed to be addressed so I will just mention the biggest and hardest difficulties to I had to overcome. Also there were some situations where the it would be impossible to test functionality because the resources were not there. An example was that there was no Windows machine currently listening for requests to run games, so i had to work around that.

3.1 Diferent Servers

This problem exists because the website were the user would submit their source code was hosted in a Linux Machine and the source code could only be run in a Windows machine by the simple fact that the game was programmed with a library (Microsoft XNA) that only runs a Windows machine. This is a huge setback because the code needs to be sent from Linux to Windows to run and then the results needed to be sent back to Linux because that was where the database with the results of all the users was located.

3.2 Efficient and effective file detection

The software that i needed to make needed to always be watching for a new user submission from the web site, and i needed, not only an efficient system, but and effective system as well. Making a system that uses as little resources as available and works every time as expected is hard and a lot of man hours went into designing the architecture for it.

• *Edgar Santos, nr. 64753,*
E-mail: edfil221@gmail.com,
Instituto Superior Técnico, Universidade de Lisboa.

incomplete: other groups members?

Manuscript received June 6, 2015.

(1.0) Excellent	LEARNINGS						DOCUMENT						
(0.8) Very Good	Context×2	Skills×1	Reflect×4	Summ×.5	Concl×.5	SCORE	Struct ×.25	Ortog×.25	Exec×4	Form ×.25	Titles ×.5	File ×.5	SCORE
(0.6) Good	0.6	0.6	0.6	0.6	0.6		0.8	0.6	0.6	0.8	0.8	1.0	
(0.4) Fair													
(0.2) Weak													

3.3 Automatic compilation and running

This was the problem where the solution was harder for me to find because in all the works I have made, I never had to manually compile a project with different components and then run the project to get the desired results. To achieve this i needed to link the project submitted by the user with the game project, then compile the entire thing. If the compilation was unsuccessful, a error message needed to be sent to the user, and if it was successful the project needed to be executed and then, when it eventually finished, send the result to the result database. So there were a lot of paths to watch and deal with.

4 The Solutions

Most of the problems where solved, but because the dimension of the project was so big, the system could be entirely finished leaving some key components just semi-built.

4.1 Diferent Servers

This task constantly needed a good communication with the other developers in the project while the development because the data structures in either side of the servers needed to be compatible with what they were doing. For example, the file submission name conventions and location needed to be known because when we eventually sent those file to another server, the server needs to be watching for files with the same naming conventions. So I needed to manage that part of the project by sending mails to both the developers of each server so that we could agree on the terms for the communication between servers. After the terms and file structure was defined I needed to define how to send the files from one server to another and then back. Due to the fact that the servers were limited to FTP, that made my work easier because I didn't had to explore the different communication protocols. After some research I re-learned how FTP worked and implemented the communication.

4.2 Efficient and Effective file detection

I took a different approach to solve this problem. Since I depended highly on this functionality for the rest of the project, I made a quick prototype that not efficient but was effective. Basically it was something that would check the contents of a folder every X seconds and then reported the new files that needed to be sent. This technique is rather useful is want something ready fast, and when you need to rework on the project to improve it, you already know what to do to improve because you already have a pretty big idea on what needs to be done. Before my second pass on this system, I searched on the Internet for systems that could track file creations, and not long after I found out that Java already had systems implemented to detect file creations as soon as they are created. This made me change the project development to Java, but since everything was still in a prototype phase, I didn't lose much time. Changing the project to Java also simplified the solution on both servers because Java is cross-platform, so what I would develop to detected new files in Linux could, at the same time, detect new files in Windows.

4.3 Automatic compilation and Running

Because of time constraints this part of the project was incomplete, but a solid foundation was built, so in future iterations, it will all be easier to fully automate the file submissions and responses.

5 Conclusion

Al tho the project wasn't 100% functional in the end, I am happy with the work I have made. I have learned many new things like scripting languages in both Windows and Linux, and also leaned about Java libraries that communicate with the file system to monitor folders and files. I was also able remember and improve what I previously learned before and then forgotten. I probably still need to have a better communication with team members, especially team members from different projects that still depend on each other.

ACKNOWLEDGMENTS

I would like to thank Teacher Rui Prada for the opportunity to work on this project and also my fellow colleagues for the patience to explain me what was previously done because I started late and needed time to catch up.