Howard Cheng, David Cross, Noemie Nakamura, John Pickering
CSCE 315-Section 502

I. Changes to Design

In the original design document, the team broke the structures used in this project down into several levels of abstraction, the highest level containing only the parser, the management system, and the application. At the higher levels of abstraction, the project ended up following the design documents exactly. For example, by the end of the project the system contained a parser, a management system, and an application. However, as the levels became less abstract, so the project started to vary from the original design. For example, in the original design for the parser, it was stated that the parser would have three separate parts: a scanner, a tokenizer, and an evaluator. In the final design of the project, the parser had those three parts, but the scanner was only a small part while the tokenizer was larger part and the evaluator even more so, instead of all three parts being equal as mentioned in the original document.

There were several other changes made at the lower level of abstraction, especially where the naming of specific functions is concerned. In the original design document, each member of the project (Tuples, Relations, etc.) was designed to have certain functions. However, because the design document was made before a single line of code was written, these functions changed and were added to as the project continued. Thus, the functions in the final version of the project are very different from those originally detailed in the design document. In addition, the functions and members of the application were described in detail in the design document, but these were similarly changed and added to as the project was completed, so that the functions and members of the application design were very different from the original design.

II. Issues Encountered

There were several major issues encountered, along with a smaller amount of immediately resolvable issues. One of the largest issues the team had was the translation between compilers in the testing of the code. This was an issue because the code the team put together immediately compiled in the Windows Visual Studio IDE, however it did not compile on the TAMU linux servers without issue. This was resolved by editing some of the code and moving files to make the linux compiler read it properly. In essence, the logic of the code was still the same, but some of the files had to be moved in order to work properly.

Another major issue encountered by the team whilst working on this project was the workload held by the other members of the team. For example, some members are taking another programming heavy course or a project-based course, meaning they did not have as much time to work on this project. As a result, the team found it difficult to keep to the milestone submissions required in the project. This issue was ultimately resolved by the team members deciding to use Skype as a form of text chat and voice chat in order to resolve issues with the project. In this way, the team members did not have to waste time and resources traveling somewhere to meet up and if they were not immediately needed in the project, they could work on other work and still be online in Skype in case they were needed again.

A third major issue encountered by the team was in communication of the specific requirements of the project at various stages. This can mainly be attributed to the requirements listed not necessarily being specific enough for the implementation of the project and to the fact that some requirements of the project were inaccurately communicated in the original assignment and were thus changed partway through the project. Though each of these issues was small and easily traversed, because there were several instances of this kind of issue, it is worth mentioning as a major issue.

III. Lessons Learned

Our team learned a lot about working in a team of more than two people, as well as many lessons in being an efficient team in general. These are most easily summarized in a list, as they don't require much individual explanation. Lessons learned were:

1. Always make sure that code compiles before you commit to the master branch

2. Test your code before your submit the final version to the master branch so that others don't encounter as many errors when they are using it.

3. Make sure to comment your code thoroughly and explain what each part does in comments. This reduces the amount of time that other members spend trying to figure out what your code does.

4. If you are making major changes to large amounts of code, work in a branch so that you don't overwrite lots of master branch code.

5. Always inform your teammates of what you are working on to avoid multiple people attempting to fix the same code/ write the same functions, etc.

6. Although assigning individual functions and tasks to team members is fine, it is prudent to make sure that no one team member is doing a large part of the code. This prevents two things: the entire project resting on the shoulders of one team member and the rest of the team having no idea how to fix large amounts of code in the project because they didn't work on it.

7. Team communication is a necessary component of success for this project and for future projects. All team members must let the rest of the team know what they are working on and if they need help or if they can't complete a portion of the project. This prevents the team from having to scramble to fix an issue caused by miscommunication the night before the project is due.

IV. Work Load Distribution

<u>Howard Cheng</u>

Portion of project completed: 25%

Worked on: Testing, engine design, design document, and bug fixing

Noemie Nakamura

Portion of project completed: 25%

Worked on: Testing, parser design, design document, and bug fixing

David Cross

Portion of project completed: 25%

Worked on: Testing, parser design, design document, and bug fixing

John Pickering

Portion of project completed: 25%

Worked on: Testing, application design, engine design, design document, and bug fixing