# CS 673
# Assignment #4
# Software Project Management Plan (SPMP)

**Team Snow Crash**
Dale Earnest
Jeff Dunn
Dong Luo
Mike McWilliams

## Table of Contents

## Revision History

| Name | Date | Reason For Changes | Version(s) |
|---|---|---|---|
| Mike | 13 Oct | Initial Draft | 0.1 |
| Mike | 18 Oct | Major updates throughout | 0.2 |
| Dale | 18 Oct | Templating | 0.3 |
| Dale | 19 Oct | Revisions throughout | 0.4 |
| Dale | 20 Oct | Final Version | 1.0 |

# 1.Introduction

The project is an application that allows the user to create or use default critters and see how they might interact in a simulated world.

## 1.1 Project Scope

The project consists of a user interface, simulation engine, and file import/export subsystem. The user creates and configures custom critters, starting from default templates, and determines the size of the simulated world. Critters are placed in the world according to the user's specifications (or a pre-determined default) when the simulation is started. Following the simulation, the user is presented with the results of the simulation.

During critter-configuration time, the user can allot scores from a points pool to each of various attributes. These attributes determine how the critter interacts with the world and other critters. At any time during this phase, the user may choose to export or import critters.

When the user is satisfied with his critters, he can run the simulation. The simulation may be paused and resumed, or stopped completely, resetting the timer to zero. The simulation has elements of randomness, so the same inputs may not necessarily generate the same outputs. The user may save the state of the world and its critters during a simulation and open that simulation at a later date.

When the simulation is over, the user is presented with a summary of results. The user may save simulation results and open them at a later date. Analysis of results is left to the user.

## 1.2 Major software functions

The software will consist of five categories, each divided into three parts. The five categories are the graphical user interface (GUI), the data store, the service layer, the world engines, and file input and output (I/O). The three parts are the three ways the user can interact with the software: configuration, simulation, and results.

## 1.3 Performance/Behavior issues

The software will iterate over the entire world every "turn", so performance must adequately support that. Turn length is limited by execution time, defaulted to one second. The time engine will repeatedly sleep for a period of time equal to turn length then fire off a tick event. The turn length may be adjusted by the software to account for the actual execution time of a turn.

## 1.4 Management and technical constraints

There will be a series of deadlines for code: initial review, formal review, final review, and delivery. The initial review date will be set internally for the purpose of intra-team review to catch and correct obvious and/or major flaws that might exist before submitting the code for formal review. Developers will be given a chance to correct that code, but not add additional code, before the formal review.

After the formal review, developers may correct any code and add additional code, provided the additional code does not significantly alter the product. A final intra-team review will precede the delivery date to ensure that the product meets specifications and will perform appropriately when delivered.

# 2.Project Estimates

This section provides cost, effort and time estimates for the projects.

## 2.1Historical data used for estimates

As this is a newly assembled team, there is no directly relevant history upon which to base an estimate. Estimates are projected by the project manager based on experience working on other projects with other teams.

## 2.2Estimation techniques applied and results

For purposes of simplicity, SLOC will be used in conjunction with the COCOMO algorithm for determining estimates.

The result is an estimate of two months of work for four developers at a rate of 21 hours per week. This scheduling poses a risk that will addressed later in the document.

### 2.2.1Estimated SLOC with COCOMO 1

Cumulatively we expect an estimated SLOC count on the order of 2,000 SLOC, where SLOC is measured as all meaningful statements, definition statements, conditional statements, other business logic, and comments. We recognize the shortcomings of this method in determining the amount of time and effort put into the project, but we feel it provides a simple way of estimating the size of a project based on its scope.

Using the basic COCOMO algorithm, our estimates are initially computed as follows:

effort = 2.4 * (KLOC=2)^1.05 = 4.97 man-months

time = 2.5 * (effort=4.97)^0.38 = 4.6 months

resources = (effort=4.97) / (time=4.6) = 1.08 developers

Note that this technique assumes each developer is working a 40 hour week.

Since it is not a viable requirement for all team members to work 40 hours per week, it is necessary to translate that figure into an estimate of hours-per-week for each developer in the four-member team:

1.08 developers * 40 hours = 4 developers * X hours

X = 10.8

So each member of the team would plan on an estimated 10.8 hours per week over a four to five month period. Because the project must be delivered in roughly two months, an estimate of 21 hours per week is more accurate. This number seems a bit steep, and could pose a potential scheduling risk. This risk will be addressed later in the document.

### 2.2.2Function Point Analysis with COCOMO 1

Using function point analysis, we developed this chart to determine the number of function points for our project.

| | **Simple** | | **Medium** | | **Complex** | | | **Sub-Total** | **Total** |
|---|---|---|---|---|---|---|---|---|---|
| | Count | Factor | Count | Factor | Count | Factor | | | |
| **Ext Inputs** | 1 | 3 | 1 | 4 | 0 | 6 | | 7 | |
| *Notes* | File I/O | | User Input | | | | | | |
| **Ext Outputs** | 2 | 4 | 0 | 5 | 1 | 7 | | 15 | |
| *Notes* | Log File, Export File | | | | User Interface | | | | |
| Ext Inq | 0 | 3 | 0 | 4 | 0 | 6 | | 0 | **49** |
| | None | | | | | | | | |
| Int Logic | 1 | 7 | 2 | 10 | | 15 | | 27 | |
| | Critters | | State, Sim Engine | | | | | | |
| Ext Interface | 0 | 5 | 0 | 7 | 0 | 10 | | 0 | |
| | None | | | | | | | | |

Using the Function Point Languages table found on the website "http://www.qsm.com/?q=resources/function-point-languages-table/index.html" we assumed that this project was of average complexity and used the number 55 as our multiplier:

49 x 55 = 2695 SLOC

Using the basic COCOMO algorithm, our estimates are initially computed as follows:

effort = 2.4 * (KLOC=2.7)^1.05 = 6.81 man-months

time = 2.5 * (effort=6.81)^0.38 = 5.2 months

resources = (effort=6.81) / (time=5.2) = 1.31 developers

Note that this technique assumes each developer is working a 40 hour week. Since that is not a viable requirement, it is necessary to translate that figure into an estimate of hours-per-week for each developer in the four-member team:

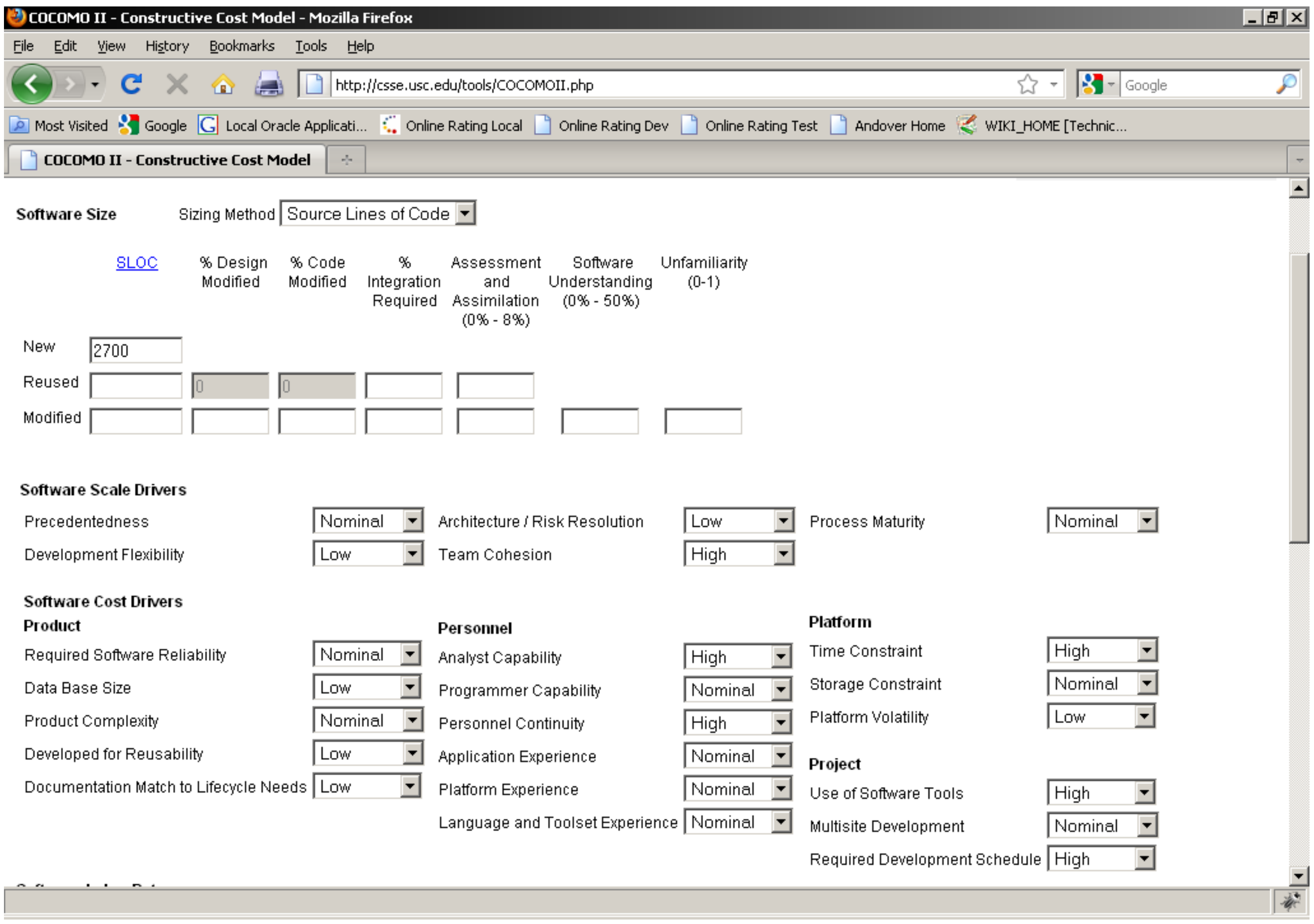
1.31 developers * 40 hours = 4 developers * X hours

X = 13.14

So each member of the team would plan on an estimated 13.14 hours per week over a four to five month period. Because the project must be delivered in roughly two months, an estimate of 26 hours per week is more accurate. This number seems a bit steep, and could pose a potential scheduling risk. This risk will be addressed later in the document.

### 2.2.3Function Point Analysis with COCOMO 2

We used the website found at "http://csse.usc.edu/tools/COCOMOII.php" to generate our COCOMO 2 estimates. These are our assumed factors that go into the equation:


COCOMO II - Constructive Cost Model - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://csse.usc.edu/tools/COCOMOII.php
Google
Most Visited Google Local Oracle Applicati... Online Rating Local Online Rating Dev Online Rating Test Andover Home WIKI_HOME [Technic...
COCOMO II - Constructive Cost Model

**Software Size** Sizing Method: Source Lines of Code

| | SLOC | % Design Modified | % Code Modified | % Integration Required | Assessment and Assimilation (0% - 8%) | Software Understanding (0% - 50%) | Unfamiliarity (0-1) |
|---|---|---|---|---|---|---|---|
| New | 2700 | | | | | | |
| Reused | | 0 | 0 | | | | |
| Modified | | | | | | | |

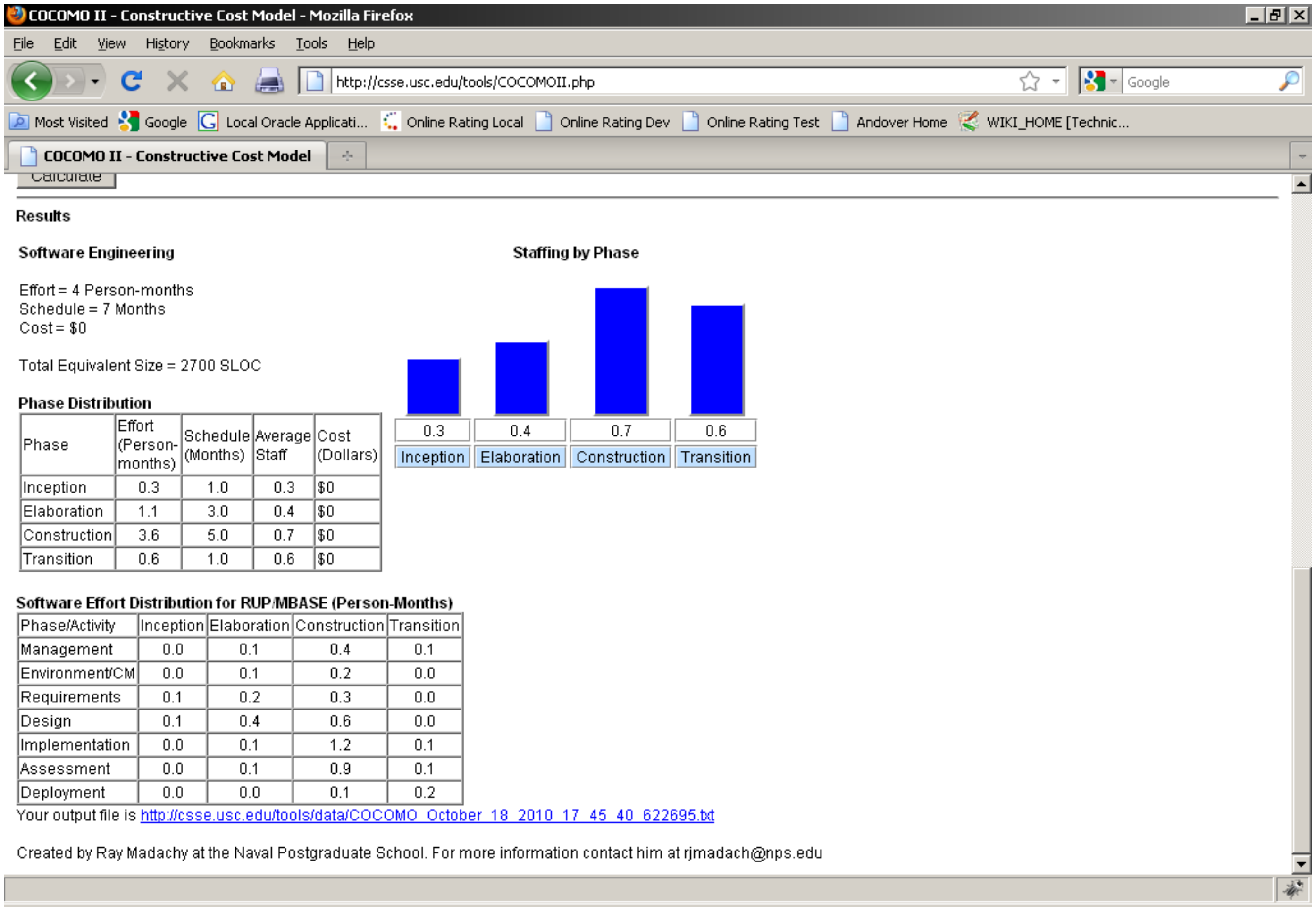
**Software Scale Drivers**

| | | | | | |
|---|---|---|---|---|---|
| Precedentedness | Nominal | Architecture / Risk Resolution | Low | Process Maturity | Nominal |
| Development Flexibility | Low | Team Cohesion | High | | |

**Software Cost Drivers**

| **Product** | | **Personnel** | | **Platform** | |
|---|---|---|---|---|---|
| Required Software Reliability | Nominal | Analyst Capability | High | Time Constraint | High |
| Data Base Size | Low | Programmer Capability | Nominal | Storage Constraint | Nominal |
| Product Complexity | Nominal | Personnel Continuity | High | Platform Volatility | Low |
| Developed for Reusability | Low | Application Experience | Nominal | **Project** | |
| Documentation Match to Lifecycle Needs | Low | Platform Experience | Nominal | Use of Software Tools | High |
| | | Language and Toolset Experience | Nominal | Multisite Development | Nominal |
| | | | | Required Development Schedule | High |

These are the results given the above factors. The results are in line with our COCOMO 1 results:


COCOMO II - Constructive Cost Model - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://csse.usc.edu/tools/COCOMOII.php
Most Visited Google Local Oracle Applicati... Online Rating Local Online Rating Dev Online Rating Test Andover Home WIKI_HOME [Technic...
COCOMO II - Constructive Cost Model
Calculate
Staffing by Phase
0.3
0.4
0.7
0.6
Inception
Elaboration
Construction
Transition

Results

Software Engineering

Effort = 4 Person-months
Schedule = 7 Months
Cost = $0

Total Equivalent Size = 2700 SLOC

Phase Distribution

| Phase | Effort (Person-months) | Schedule (Months) | Average Staff | Cost (Dollars) |
|---|---|---|---|---|
| Inception | 0.3 | 1.0 | 0.3 | $0 |
| Elaboration | 1.1 | 3.0 | 0.4 | $0 |
| Construction | 3.6 | 5.0 | 0.7 | $0 |
| Transition | 0.6 | 1.0 | 0.6 | $0 |

Software Effort Distribution for RUP/MBASE (Person-Months)

| Phase/Activity | Inception | Elaboration | Construction | Transition |
|---|---|---|---|---|
| Management | 0.0 | 0.1 | 0.4 | 0.1 |
| Environment/CM | 0.0 | 0.1 | 0.2 | 0.0 |
| Requirements | 0.1 | 0.2 | 0.3 | 0.0 |
| Design | 0.1 | 0.4 | 0.6 | 0.0 |
| Implementation | 0.0 | 0.1 | 1.2 | 0.1 |
| Assessment | 0.0 | 0.1 | 0.9 | 0.1 |
| Deployment | 0.0 | 0.0 | 0.1 | 0.2 |

Your output file is http://csse.usc.edu/tools/data/COCOMO_October_18_2010_17_45_40_622695.txt

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu

## 2.3Project Resources

The developers for the project are: Dale Earnest, Jeff Dunn, Dong Luo, and Mike McWilliams.

No specific hardware resources are needed.

Developers are expected to use Eclipse for development purposes. Developers must have the Java Development Kit (JDK) version 1.6 in order to author and run the application. JSON must be added to source control and included as a project dependency, if it is to be used for file I/O.

Configuration management is described in the SCMP.

# 3.Project Schedule

The project tasks can best be broken down into the five categories described in section 1.2: GUI, data store, service, engines, and file I/O. The project will be broken down into three phases. The GUI, services, and file I/O tasks have subtasks in each of the three phases. The data store will be fully implemented by the end of phase one. The engines will be fully implemented by the end of phase two.

## 3.1Project task set

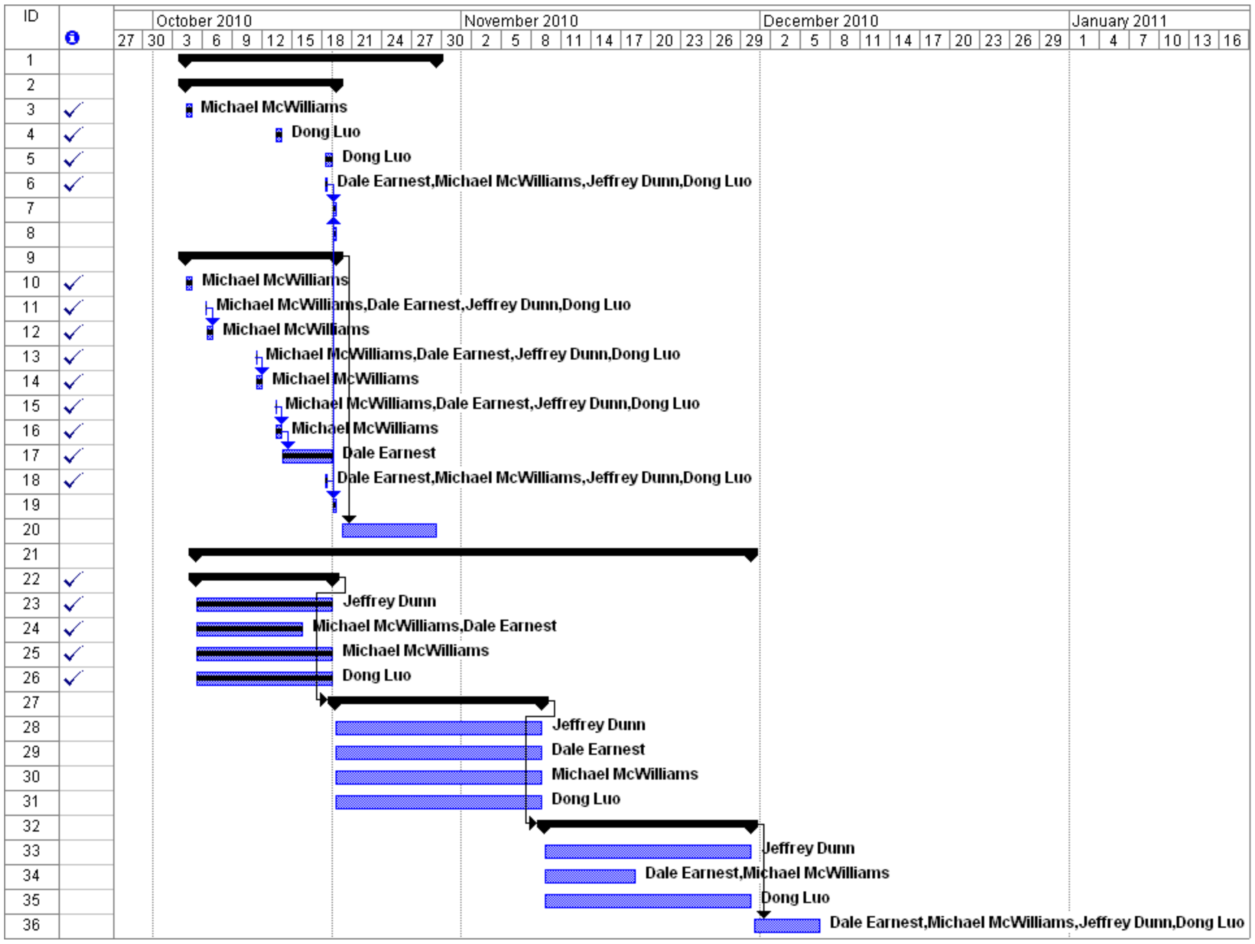
The project tasks are: GUI, data store, service, engines, and file I/O.

## 3.2Functional decomposition

- GUI: This is the graphic user interface, comprising all the user activity and graphical interactions that the application will possess.
- Data Store: This contains the necessary data and logical information upon which the Engine and GUI determine how they act.
- Service: The service layer managed the interactivity between the UI, Engine, Data Store, and File I/O.
- Engine: The Engine determines the actions of the Critters and how they interact with each other in the World map.
- File I/O: The portion manages the import and export of configuration files and saved simulations.

## 3.3Timeline chart

Below is our project timeline Gantt chart.



# 4.Risk Management

This section discusses project risks and the approach to managing them.

## 4.1Project Risks and Mitigation

### 4.1.1Loss of code repository

It is possible, however unlikely, that we could temporarily or permanently lose our code repository.

We have not developed a solution for this, since, we believe, the likeliness of this event occurring is minimal.

#### 4.1.2JSON bugs

It is possible that we could encounter bugs with the JSON utility software.

If this event occurs, we will revert to a different mechanism for file input and output, likely using the default Java libraries.

#### 4.1.3Weather

Weather could interfere with team meetings; the end of the project overlaps with winter and as half of our team members travel far to get to team meetings and class, we will have to be aware of inclement weather.

Adjustments will be made on a case-by-case basis, depending on how vital the meeting is and how soon the team leader is alerted. Possible solutions could include teleconferencing and holding the meeting with a partial team, sending out meeting minutes, and communicating concerns by email.

#### 4.1.4Less-experienced Java developers

While two of us are experienced Java programmers, one of us has just recently started learning Java and another has no prior Java experience. There is a particularly large risk with Swing, with which none of us is terribly familiar.

To address this risk, the core functionality is being developed by the two developers who are more familiar with Java. They will be able to support the other two developers as necessary. Additionally, the two less-experienced Java developers will be attempting to learn the relevant parts of the language. One of the latter two will be focusing on Swing, allowing him to focus his resources on the GUI. The other developers will be available to delegate work to.

#### 4.1.5Excessive scope in requirements

Excessive requirements increase the scope of the project, thereby increasing the amount of time and effort that must be spent in completing the project.

To address this issue, "nice to have" functionality, or gold-plating, are relegated to lower-priority requirements that may be addressed, time-permitting, after all other functionality has been completed.

#### 4.1.6Schedule slip due to external life factors

External life factors, including those related to work, family, and health, could delay project progression and interfere with team meetings.

These issues are largely unpredictable, but we will try to account for them in our projections, set earlier-than-necessary due dates, and cooperate, on the fly, to complete all functional areas.

### 4.1.7Rework due to faulty design

Design is not an exact science, and an architecture that looks appropriate at the beginning of a project may need to be revisited as the project progresses. This often involves rework of code that has already been written.

Our design decisions will be made to encourage modularity and extensibility. This should allow us to swap different functional areas out, and add additional functionality, with limited changes to existing code.

### 4.1.8Changed or added requirements

Changed and added requirements can delay project progression and add to the amount of rework involved in the project.

Any non-essential requirements that are added after the SRS is base-lined will be considered gold-plating and will be given a low priority. Time-permitting, we may revisit these requirements and implement them if they definitively add to the user experience.

After the requirements have been base-lined, any changes will be added as new requirements. Excepting special cases (decided by the team leader), development will continue using the base-lined requirements.

## 4.2 Risk Table

[Note: highest priority is lowest number]

| Risk No. | Risk Description | Likelihood to occur L (1-10) | Impact I (1-10) | Retirement Cost R (1-10) | Priority (11-L)*(11-I)*R | Owner | Completion Target Date |
|---|---|---|---|---|---|---|---|
| 1 | Loss of code repository | 1 | 10 | 8 | 80 | Dale Earnest | Within 1 week of notification |
| 2 | JSON bugs | 2 | 3 | 8 | 576 | Dong Luo, Dale Earnest | Within 1 week of notification |
| 3 | Weather | 8 | 4 | 4 | 84 | Dale Earnest, team | 24 hours |
| 4 | Experience | 10 | 6 | 9 | 45 | Dong Luo, Jeff | 11/17/10 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | Dunn, team | |
| **5** | Excessive requirements | 7 | 7 | 2 | 32 | Dale Earnest, Mike McWilliams, team | 11/03/10 |
| **6** | External life factors | 10 | 4 | 4 | 28 | Dale Earnest, team | Within 2 days |
| **7** | Faulty design | 3 | 7 | 7 | 224 | Mike McWilliams | 11/17/10 |
| **8** | Changes to SRS | 7 | 7 | 2 | 32 | Dale Earnest, Mike McWilliams, team | 11/17/10 |

### 4.3Risk Monitoring and Management

Risks will be addressed as the events that cause them occur, unless otherwise indicated in the chart above. Each team member will have the chance at team meetings to bring up any encountered events that may add additional risk to the project. The team will discuss each risk presented and a decision will be reached, with the team leader ultimately having the final say.

## 5.Staff Organization

The manner in which staff is organized and the mechanisms for reporting are noted.

### 5.1Team structure

The team leader is Dale Earnest. As such, he has the final say in all managerial, design, code, etc., decisions. Dale will also be in charge of the data portion of the project and will take part in developing the engines.

Jeff Dunn is the head GUI developer, in charge of creating screen mock-ups, holding screen reviews, and developing the screens of the user interface.

Dong Luo is developing the file I/O portion of the project and will determine the format of the different output files.

Mike McWilliams is the head architect. As such, his responsibility is to design the system in such a way that all developers can code their respective pieces simultaneously. Mike is also in charge of the services and will take part in developing the engines. He will also oversee the overall development effort and work on design decisions with the other developers.

### 5.2Management reporting and communication

All general issues will be raised to the entire group so they can be resolved quickly. Specific issues related to design may be addressed directly to the head architect, or to the group as a whole. Specific issues related to the overall path of the project, the requirements of the project, or managerial decisions may be addressed directly to the team leader, or to the group as a whole. Sensitive issues, should they arise, should be addressed directly, and only, to the team leader.

The team leader has the final say on all matters related to the project.

## 6.Tracking and Control Mechanisms

Project tracking is the responsibility of the team leader. Once we enter the Requirements Analysis phase of the project, status will be reported weekly directly to the team leader before the weekly meeting. Status should include any major milestones achieved, any blocking issues, and any risks that have been uncovered. The team leader will review all important status during the meeting and may allocate additional resources to an area, if needed.

The team leader reserves the right to de-scope certain parts of the project, and to schedule arbitrary interim builds.

## 7.Google Code Repository

Our Google Code repository can be found at: http://code.google.com/p/teamsnowcrash/.

# 8.Hours

## 8.1Total Hours

| Deliverable | Estimated Hours | Actual Hours |
|---|---|---|
| STCD | 30 | 25 |
| SPD | 40 | 38 |
| SCMP | 40 | 57 |
| SPMP | 40 | 62 |
| SRS | 60 | |
| SDD | 40 | |
| Group Project Presentation | 30 | |
| Individual Project Report | 10 | |
| Totals | 290 | 182 |

## 8.2Hours Broken Down

| Item | Dale Earnest | Mike McWilliams | Jeff Dunn | Dong Luo | Total Hrs. |
|---|---|---|---|---|---|
| Meetings | 6 | 6 | 6 | 6 | 24 |
| Communications Plan | 0 | 0 | 0 | 0 | 0 |
| Application Investigation | 1 | 1 | 12 | 4 | 18 |
| Prototype (coding) | 1 | 0 | 0 | 0 | 1 |
| Prototype (drawings) | 1 | 1 | 0 | 0 | 2 |
| Project Proposal | 0 | 0 | 0 | 0 | 0 |
| Document Editing | 2 | 8 | 0 | 3 | 13 |
| Document Review | 1 | 1 | 1 | 1 | 4 |
| Total Hours | 12 | 17 | 19 | 14 | 62 |