Project Plan

TEAM 1: CURTIS BROWNN
JACOB DAVIDSON
BRANT MULLINIX

CSC 478 – B
NOVEMBER 29, 2014

Project Plan – Table of Contents

Scope Statement

1. Proposed Project Summary

For the semester project, team 1 proposes to develop a version of the popular dice game Farkle. Our proposed version will be a standalone application utilizing a graphic user interface. The first iteration of development will yield a one player version of the game, and two player support will be added in a second iteration with the second player being a live person using the same computer as the first player. If time permits, a third iteration will yield a two player version that can be played against a computer opponent with probability determining the play of the computer opponent.

2. Traditional Rules of Farkle

Farkle is a game typically played with six dice and more than one player. The traditional rules dictate that each player takes turns rolling the dice in succession, with each turn producing a score. The score produced from the players current turn is added to that players previous score accumulation. The goal is to be the first player to reach 10,000 points. The scoring is generated as follows:

1. At the beginning of the turn, the player rolls all 6 dice.
2. Scoring for each roll is as follows: Each 1 = 100 points, each 5 = 50 points, (3) 1s = 1000 points, (3) 2s = 200 points, (3) 3s = 300 points, (3) 4s = 400 points, (3) 5s = 500 points, (3) 6s = 600 points, a straight = 1500 points, and more than three of a kind doubles the value for each additional match (e.g. (5) 3s = 300 x 2 x 2 = 1200 points).
3. Dice resulting in a score are chosen and removed by the player (the player must pick up at least one scoring die, but does not need to pick up all scoring die), and the player decides if they want to roll with the remaining dice or pass to the next player.
4. At least one die must be set aside after each roll.
5. A “farkle” occurs when a roll results in no points. All points accrued during that turn are forfeit, and play is passed to the next player.
6. If the player has scored all six dice, he or she can roll again with all six dice.
7. Once a player has reached the winning point total, each successive player has one last chance to score enough points to surpass the leader.

3. Proposed Changes to the Traditional Rules for this Version of the Game

Traditionally, Farkle is a multiplayer game with the first player to a given point total being named the winner of the game. The primary play for our proposed version will center on a single player. Two player options will be available in the final release, but we anticipate most players will opt for single player gameplay making it the primary focus of this application. To differentiate single player play from multiplayer play, the single player version will limit the number of turns given to the player to ten with the primary goal being to maximize the total points for each game. In a sense, the player will be playing against himself or herself trying to beat a previous high score.

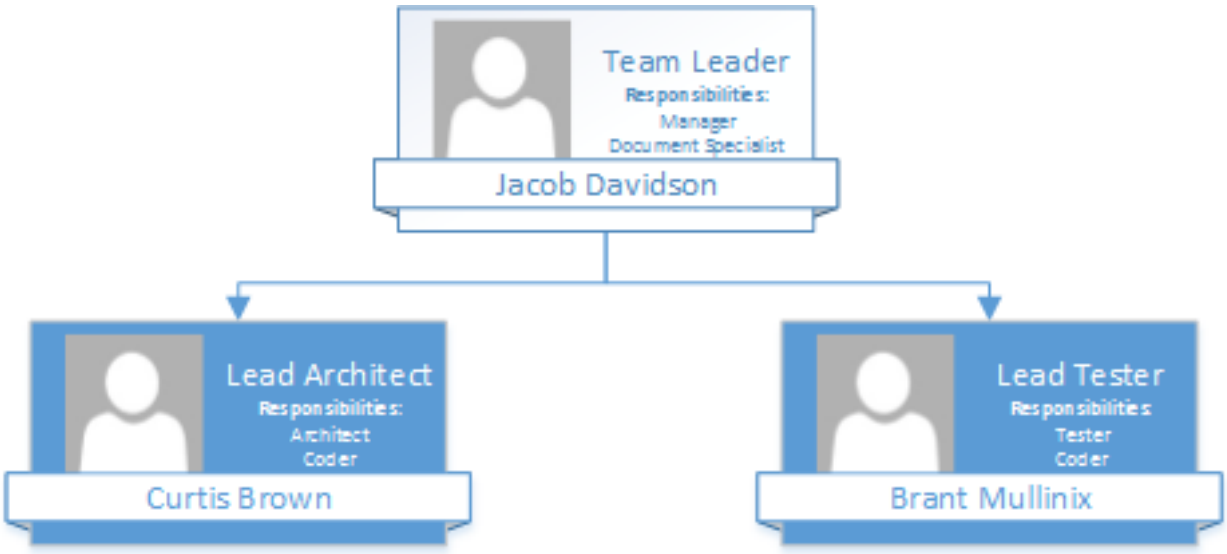
4. Programming Language

The application will be written in Java using JDK version 7.

5. Platform and Software Requirements

The application will be developed and tested on the Microsoft Windows 7 platform with the Java Runtime Environment 1.7+ installed.

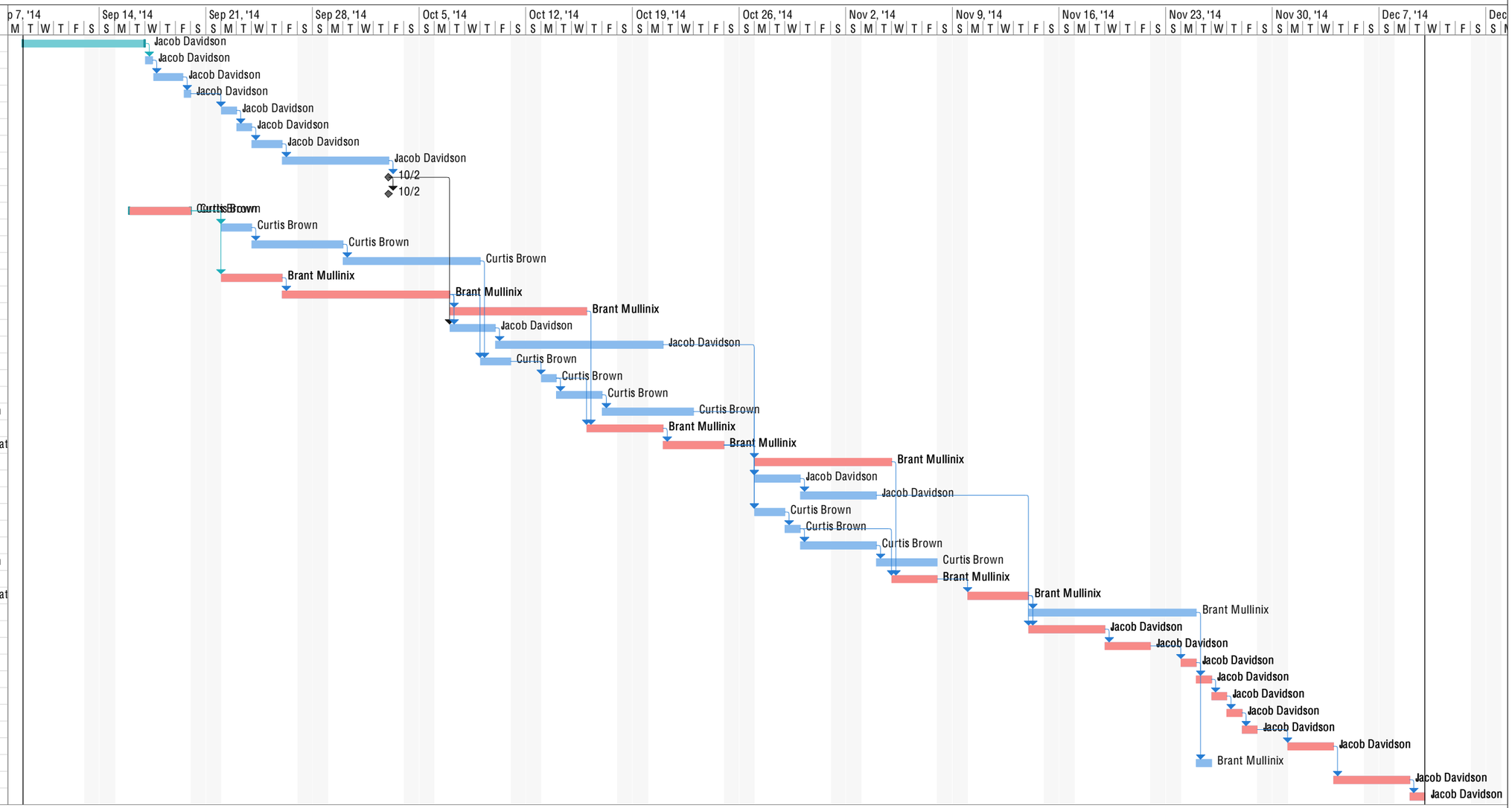
Organizational Chart



Team Structure: Democratic Centralized

ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Project Plan**	**10 days**	**Tue 9/9/14**	**Mon 9/22/14**		
2	Scope Statement	6 days	Tue 9/9/14	Tue 9/16/14		Jacob Davidson
3	Org Chart	0.5 days	Wed 9/17/14	Wed 9/17/14	2	Jacob Davidson
4	Gantt Chart	2 days	Wed 9/17/14	Fri 9/19/14	3	Jacob Davidson
5	Tools and Standards	0.5 days	Fri 9/19/14	Fri 9/19/14	4	Jacob Davidson
6	Configuration Management Plan	1 day	Mon 9/22/14	Mon 9/22/14	5	Jacob Davidson
7	**Programmer's Manual**	**57 days**	**Tue 9/16/14**	**Wed 12/3/14**		
8	**Requirements Documentation**	**50 days**	**Tue 9/16/14**	**Mon 11/24/14**		
9	Introduction	1 day	Tue 9/23/14	Tue 9/23/14	6	Jacob Davidson
10	General Description	2 days	Wed 9/24/14	Thu 9/25/14	9	Jacob Davidson
11	Specific Requirements	5 days	Fri 9/26/14	Thu 10/2/14	10	Jacob Davidson
12	Appendices (None)	0 days	Thu 10/2/14	Thu 10/2/14	11	Jacob Davidson
13	Index (Don't Include)	0 days	Thu 10/2/14	Thu 10/2/14	12	Jacob Davidson
14	**1st Code Iteration**	**25 days**	**Tue 9/16/14**	**Mon 10/20/14**		
15	Architecture Diagram (UML)	4 days	Tue 9/16/14	Fri 9/19/14		Curtis Brown
16	Decision Tables/Control Diagrams	2 days	Mon 9/22/14	Tue 9/23/14	15	Curtis Brown
17	GUI Pseudocode	4 days	Wed 9/24/14	Mon 9/29/14	16	Curtis Brown
18	GUI Code/Source Code Documentation	7 days	Tue 9/30/14	Wed 10/8/14	17	Curtis Brown
19	Backend Pseudocode	4 days	Mon 9/22/14	Thu 9/25/14	15	Brant Mullinix
20	Backend Code/Source Code Documentation	7 days	Fri 9/26/14	Mon 10/6/14	19	Brant Mullinix
21	Testing/Testing Documentation	7 days	Tue 10/7/14	Wed 10/15/14	20	Brant Mullinix
22	Update Requirements	3 days	Tue 10/7/14	Thu 10/9/14	20,12	Jacob Davidson
23	User's Manual	7 days	Fri 10/10/14	Mon 10/20/14	22	Jacob Davidson
24	**2nd Code Iteration**	**19 days**	**Thu 10/9/14**	**Tue 11/4/14**		
25	Update Architecture Diagram (UML)	2 days	Thu 10/9/14	Fri 10/10/14	18,20	Curtis Brown
26	Update Decision Tables/Control Diagrams	1 day	Mon 10/13/14	Mon 10/13/14	25	Curtis Brown
27	Update GUI Pseudocode	3 days	Tue 10/14/14	Thu 10/16/14	26	Curtis Brown
28	Update GUI Code/Source Code Documentation	4 days	Fri 10/17/14	Wed 10/22/14	27	Curtis Brown
29	Update Backend Pseudocode	3 days	Thu 10/16/14	Mon 10/20/14	21,26	Brant Mullinix
30	Update Backend Code/Source Code Documentation	4 days	Tue 10/21/14	Fri 10/24/14	29	Brant Mullinix
31	Testing/Testing Documentation	7 days	Mon 10/27/14	Tue 11/4/14	30	Brant Mullinix
32	Update Requirements	3 days	Mon 10/27/14	Wed 10/29/14	30,23	Jacob Davidson
33	Update User's Manual	3 days	Thu 10/30/14	Mon 11/3/14	32	Jacob Davidson
34	**3rd Code Iteration**	**21 days**	**Mon 10/27/14**	**Mon 11/24/14**		
35	Update Architecture Diagram (UML)	2 days	Mon 10/27/14	Tue 10/28/14	28,30	Curtis Brown
36	Update Decision Tables/Control Diagrams	1 day	Wed 10/29/14	Wed 10/29/14	35	Curtis Brown
37	Update GUI Pseudocode	3 days	Thu 10/30/14	Mon 11/3/14	36	Curtis Brown
38	Update GUI Code/Source Code Documentation	4 days	Tue 11/4/14	Fri 11/7/14	37	Curtis Brown
39	Update Backend Pseudocode	3 days	Wed 11/5/14	Fri 11/7/14	36,31	Brant Mullinix
40	Update Backend Code/Source Code Documentation	4 days	Mon 11/10/14	Thu 11/13/14	39	Brant Mullinix
41	Testing/Testing Documentation	7 days	Fri 11/14/14	Mon 11/24/14	40	Brant Mullinix
42	Update Requirements	3 days	Fri 11/14/14	Tue 11/18/14	40,33	Jacob Davidson
43	Update User's Manual	3 days	Wed 11/19/14	Fri 11/21/14	42	Jacob Davidson
44	**Finalize Design Documentation**	**8 days**	**Mon 11/24/14**	**Wed 12/3/14**		
45	Architecture Diagram Review	1 day	Mon 11/24/14	Mon 11/24/14	43	Jacob Davidson
46	Pseudocode Review	1 day	Tue 11/25/14	Tue 11/25/14	45	Jacob Davidson
47	Decision Tables Review	1 day	Wed 11/26/14	Wed 11/26/14	46	Jacob Davidson
48	Control Diagrams Review	1 day	Thu 11/27/14	Thu 11/27/14	47	Jacob Davidson
49	Source Code Documentation Review	1 day	Fri 11/28/14	Fri 11/28/14	48	Jacob Davidson
50	Testing Documentation Review	3 days	Mon 12/1/14	Wed 12/3/14	49	Jacob Davidson
51	Known Bugs and Issues	1 day	Tue 11/25/14	Tue 11/25/14	41	Brant Mullinix
52	**User Documentation**	**4 days**	**Thu 12/4/14**	**Tue 12/9/14**		
53	Finalize User's Manual	3 days	Thu 12/4/14	Mon 12/8/14	50	Jacob Davidson
54	Installation Instructions	1 day	Tue 12/9/14	Tue 12/9/14	53	Jacob Davidson

ID	Task Name
2	Scope Statement
3	Org Chart
4	Gantt Chart
5	Tools and Standards
6	Configuration Management Plan
9	Introduction
10	General Description
11	Specific Requirements
12	Appendices (None)
13	Index (Don't Include)
15	Architecture Diagram (UML)
16	Decision Tables/Control Diagrams
17	GUI Pseudocode
18	GUI Code/Source Code Documentation
19	Backend Pseudocode
20	Backend Code/Source Code Documentation
21	Testing/Testing Documentation
22	Update Requirements
23	User's Manual
25	Update Architecture Diagram (UML)
26	Update Decision Tables/Control Diagrams
27	Update GUI Pseudocode
28	Update GUI Code/Source Code Documentation
29	Update Backend Pseudocode
30	Update Backend Code/Source Code Documentat
31	Testing/Testing Documentation
32	Update Requirements
33	Update User's Manual
35	Update Architecture Diagram (UML)
36	Update Decision Tables/Control Diagrams
37	Update GUI Pseudocode
38	Update GUI Code/Source Code Documentation
39	Update Backend Pseudocode
40	Update Backend Code/Source Code Documentat
41	Testing/Testing Documentation
42	Update Requirements
43	Update User's Manual
45	Architecture Diagram Review
46	Pseudocode Review
47	Decision Tables Review
48	Control Diagrams Review
49	Source Code Documentation Review
50	Testing Documentation Review
51	Known Bugs and Issues
53	Finalize User's Manual
54	Installation Instructions



Project: Gantt Chart | Task | Manual Task | Critical

Team 1: Curtis Brown, Jacob Davidson, Brant Mullinix | Date: November 29, 2014

Tools

1. Programming Language

This application will be written in the Java programming language. Specifically, Java version 1.7 will be used via the Java Platform Tools Standard Edition 7 Development Kit provided by Oracle.

2. Integrated Development Environment

The open source Eclipse integrated development environment, available at http://www.eclipse.org/, will be used for this application. The EGit plugin for Eclipse, available at http://www.eclipse.org/egit/, will be added to the standard Eclipse installation to enable integration with the Git version control system. The M2E – Maven Integration for Eclipse plugin, available at http://www.eclipse.org/m2e/, will be used to package the application into a single executable file.

3. Version Control

All code changes will be tracked using the Git version control system, enabling collaborative coding and a central repository for the master version of the code. The master repository will be stored on Google Code at the following URL: https://code.google.com/p/farkle-csc478/. At the completion of each version of the application, the master branch will be forked, effectively saving a snapshot of the code for that version.

4. Generating the Installer

The .msi installer for this application will be created in MS Visual Studio 2012 using the InstallShield plugin made available by Flexera Software, http://www.installshield.com. The .exe created from the Maven build in eclipse will be used to create this installer.

5. Documentation Editor

All documentation will be initially drafted and maintained in Microsoft Word 2013. Collaboration on these documents will be facilitated via a shared folder on Google Drive.

6. Scheduling

Project scheduling will be completed in Microsoft Project 2013. At the conclusion of the project, the Gantt chart generated in Microsoft Project will be submitted in pdf format.

7. Uniform Modeling Language Tool

The overall architecture of the application will be modeled in Microsoft Visio 2013 using the standards of the Uniform Modeling Language. At the conclusion of the project, the UML diagram will be submitted in pdf format.

8. Online Meetings

Online meetings will be conducted via Google Hangouts twice weekly. Google Hangouts allows desktop sharing, and video conferencing among multiple users. The conference calling line owned by the

company Curtis works for will serve as a backup option if any team member is unable to use Google Hangouts for any scheduled meeting.

9. File Sharing

A shared folder has been set up on Google Docs to facilitate the sharing and storage of all files developed for the project that are not stored on Google Code.

Standards

1. Process Model

The application will be developed using the incremental process model with three increments scheduled for completion prior to submitting the final project.

2. Overall Architectural Model

The application will be developed using a variation of the model-view-controller architectural pattern. In this variation, the controller will receive input from the view, send it to the model, receive a response from the model, and update the view. In essence, the controller is responsible for all communication between the model and view.

3. Code Documentation Standards

All source code documentation will follow the Javadoc standard allowing for automatic generation of formatted source code documentation in the Eclipse integrated development environment.

Configuration Management Plan

1. Software Management

Maintaining and tracking the various versions of this software will be done using Git in conjunction with Google Code (https://code.google.com/). The master repository, which will store the most up-to-date version of the software and all committed changes to that software, will be stored at https://code.google.com/p/farkle-csc478/. Each student working on this project will download a local copy of the master repository, make changes to the code (testing the changes as they progress), and upload those changes back to the master repository. This workflow will allow each individual to test any code changes made prior to uploading revised code, and will also allow the team to easily revert to a past version of the code should mistakes be made.

Versions will be tracked using the *major.minor.build* convention. Three major releases are anticipated during the incremental approach used in developing this software, and these releases will be labeled versions 1.0.0, 2.0.0, and 3.0.0. The minor versions will be updated as bug fixes are incorporated, and the build versions will be created as needed. Upon each major release, the master repository will be forked allowing for the rapid recreation of older versions (should they be needed). In addition to forking the master repository, a copy of the entire project folder will be compressed and stored in a shared folder on Google Drive ensuring redundancy in the stored versions of the software.

2. Documentation Management

The master version of all documentation will be created in Microsoft Word 2013 and stored in a shared folder on Google Drive. Google drive will facilitate the collaboration of multiple students on any given document. Applicable documents will be saved and stored for each iteration in the development process.

Weekly Status Reports

TEAM 1: CURTIS BROWNN
JACOB DAVIDSON
BRANT MULLINIX

CSC 478 – B
NOVEMBER 29, 2014

Weekly Report – 9/10 to 9/16

1. Weekly Report Schedule

We've chosen to have meetings every Saturday and Tuesday. Weekly reports will be issued after every Tuesday's meeting.

2. Project Idea

We selected Curtis's idea of the dice game Farkle. We'll use an iterative approach starting with a one player version, adding a multiple player version, and adding a computer player to the multiple player version. We'll be using a graphic user interface for this project, and will also save past high scores in a text file. We've agreed to use Java for the language for this project.

3. Team Structure

We've selected the democratic centralized team structure. The team leader will serve as the manager for the project. In addition to organizing the meetings, this person will serve as the document specialist. Another team member will serve as the architect and the coder of the graphic user interface for the application. The final team member will serve as the tester and the coder of the back end. These will be the rolls each team member is responsible for, but each person will review and help with the other's work. The selected team structure is as follows:

- Jacob Davidson - Team Leader/Document Specialist
- Curtis Brown - Architect/GUI Coder
- Brant Mullinix - Tester/Back End Coder

4. Meeting Schedule

We've agreed to meet on Saturday mornings at 10am central (11am eastern), and on Tuesdays at 6pm central (7pm eastern) via Google Hangouts beginning on Tuesday September 16th. Brant has sent out a google calendar reminder to Jake and Curtis as a reminder for these meetings. We'll use Curtis's work conference line as a backup means of communication should one of us not be able to access Google Hangouts for a given meeting.

5. File Collaboration

For most documentation we'll use Google Drive. Brant set up a shared folder Curtis and Jake can access. For coding version control, we'll use Git along with code.google.com. We'll also use code.google.com for version control of a documentation wiki where it makes sense.

6. Project Schedule

A general layout for the project schedule was agreed upon.

7. Tasks for Next Week

The Scope Statement will be finalized and submitted. Jacob will create a rough draft of the Gantt chart that will be discussed in the next meeting, and will continue work on the rest of the Project Plan. Curtis and Brant will begin working on the application architecture which will also be discussed in the next meeting (and subsequently used in finalizing the requirements in the requirements documentation for the first iteration).

Weekly Report – 9/17 to 9/23

1. Overall Architecture

We all agreed on a conceptual view of the overall architecture. Curtis has begun fleshing out the architectural details of the application and has completed a rough draft of the UML diagram.

2. Project Schedule

Jacob developed the initial project schedule in Microsoft Project. Everyone agreed that this rough draft version of the schedule is a good starting point. The initial schedule includes three iterations of code, so we decided to complete the first iteration to determine if the schedule is realistic. After the first iteration is complete, we will revise the schedule.

3. Requirements Specification

Jacob developed and Curtis reviewed the detailed specification of the requirements document which will be used to enable traceability when Brant and Curtis are coding the first iteration, and subsequent iterations, of the application.

4. Source Code Documentation

It was agreed that the Javadoc standard would be used for source code documentation.

5. Tasks for Next Week

Curtis will finish up the overall architecture for the first code iteration. Brant and Curtis will continue working on the pseudocode/source code for the first iteration which isn't expected to be finished until October 8th. Jacob will finish the project plan and requirements documentation.

Weekly Report – 9/24 to 9/30

1. Coding - First Iteration

Brant and Curtis are on schedule to complete their tasks for the first iteration of development by October 8th. Curtis is approximately 90% done with the UML diagram for this iteration, 50% done with the pseudocode for the GUI and controller, and 75% done with the source code for the GUI and controller. Brant is 90% done with the pseudocode and 75% done with the source code for the logic.

2. Project Plan and Requirements Documentation

Jacob has finished the first iteration of the Project Plan and Requirements Documentation. The requirements documentation will be reviewed and updated as necessary after the first iteration of coding is complete.

3. Project Builds

Curtis has migrated the source code to maven to automate the build and testing process.

4. Bug tracking

We've agreed to use the issue tracking capability of Google Code to track all bugs found during testing. Beginning Monday, October 6, all bugs found will be documented in Google Code.

5. Tasks for Next Week

Brant and Curtis will continue working on the source code, UML, and pseudocode for the first iteration. Jacob will take a break from documentation to work on testing until the first iteration is complete.

Weekly Report – 10/1 to 10/7

1. Coding - First Iteration

Brant and Curtis have completed the first iteration of code, and we're ready to begin formal testing. The source code documentation still needs to be updated for traceability with the requirements specification.

2. Testing

Junit testing has begun. Curtis has begun testing the FarkleController class and the Die class. Jacob is 75% done with testing the calculateScore method.

3. Tasks for Next Week

Jake will finish the calculateScore method testing and begin black box testing. Brant and Curtis will continue white box testing.

Weekly Report – 10/8 to 10/14

1. Coding - First Iteration

The first iteration of the application has been completed. The origin repository has been forked to preserve the state of the application at the conclusion of the first iteration.

2. Testing

Brant is incorporating bug fixes, resulting from testing the first iteration, into the project. These fixes will be incorporated into the second iteration of the code.

3. Second Iteration

The team will begin working on implementing the features of the second iteration. Coding for this iteration is scheduled for completion on 10/27.

4. Tasks for Next Week

Curtis will continue white box testing the first iteration. Jake will update the requirements document for the second Iteration, and Brant will begin coding the second iteration.

Weekly Report – 10/15 to 10/21

1. Coding - Second Iteration

Brant and Curtis continue working on coding for the second iteration. Curtis has updated the GUI for a better overall look and feel. Jacob has created a custom scoring summary image that has been incorporated in the GUI.

2. Requirements Document

Jacob has updated the specific requirements of the requirements document to incorporate second and third iteration objectives.

3. Testing

Brant has incorporated most of the bug fixes resulting from black box testing of the first iteration. Jacob has completed a formal black box testing document.

4. Tasks for Next Week

Brant and Curtis will continue coding iteration 2. Jacob will open issue tickets for all bugs that arose during formal black box testing. Jacob will add traceability to the calculateScore Junit test.

Weekly Report – 10/22 to 10/28

1. Coding - Second Iteration

Brant revamped and greatly simplified the game mode selection process, and created an early implementation of two player mode. Iteration 2 is on schedule to be complete by Thursday, November 6th per the project plan.

2. Requirements Document

Jacob incorporated Brant's revamped game mode selection process into the requirements document.

3. Testing

Curtis is continuing to develop Junit tests for white box testing. Jacob issued tickets that arose during formal black box testing of the first iteration.

4. Tasks for Next Week

Brant and Curtis will continue coding and testing iteration 2, and Jacob will begin creating a user's manual for the application.

Weekly Report – 10/29 to 11/4

1. Coding - Second Iteration

Iteration 2 is complete, and the main git repository has been branched to preserve the state of iteration 2.

2. Coding – Third Iteration

Curtis will finish the end of game logic for iteration 3 before working on refactoring the code. Brant will finish the remainder of the third iteration requirements. November 24th is the planned completion date for iteration 3, but Brant has indicated he can incorporate the third iteration requirements by November 9th.

3. User's Manual

The first draft of the user's manual is currently in progress and is 50% finished.

4. Tasks for Next Week

Curtis will incorporate the end of game logic, and Brant will incorporate the rest of the third iteration requirements. Jake will finish the User's Manual, and start black box testing of the 2nd iteration.

Weekly Report – 11/5 to 11/11

1. Coding – Third Iteration

Brant completed the implementation of all third iteration requirements. Curtis currently has control of the code base and is refactoring and finalizing the third iteration.

2. User's Manual

Jacob has finished the first draft of the user's manual.

3. Testing

Jacob has finished formal black box testing of the third iteration requirements and raised all issues on Google code.

4. Tasks for Next Week

Curtis will finish refactoring and finalizing the code base by the end of Saturday, November 15th. After the 15th, Jacob and Curtis will work on white box testing while Brant corrects all issues raised on Google code.

Weekly Report – 11/12 to 11/18

1. Coding – Third Iteration

Curtis completely refactored the code base at the end of the third iteration and implemented all final application features.

2. Testing

Jacob and Curtis are completing Junit testing. Any bugs that are found will be communicated via Google Code Issues.

3. Bug fixes

Brant is working on fixing all issues that have been raised on Google Code.

4. Tasks for Next Week

Jacob and Curtis will finish JUnit testing, and Brant will continue to incorporate fixes.

Weekly Report – 11/19 to 11/25

1. Testing

Jacob has finished Junit testing the classes for which he is responsible and incorporated all associated bugs that arose. Curtis will finish Junit testing the classes for which he's responsible by the end of Friday, November 28. Any bugs that he finds will be communicated via Google Code Issues.

2. UML Document

Jacob updated the UML document for the third iteration.

3. Bug fixes

Brant is working on fixing all issues that have been raised on Google Code.

4. Tasks for Next Week

Curtis will finish Junit testing, and Brant will finish incorporating as many bug fixes as possible. Jacob will coordinate all documentation and finalize the project.

Weekly Report – 11/26 to 12/2

1. Testing

Curtis has finished Junit testing of the classes for which he was responsible.

2. Bug Fixes

Jacob updated the UML document for the third iteration.

3. Finalizing the project

Jacob finished commenting the source code, generated pseudocode for a few important methods, generated decision tables, created a control chart for the application flow, coordinated the requirements document with the source code, coordinated the requirements document with the Junit testing code, updated the user manual, finalized the formal testing document, finalized the Gantt chart, created the known bugs and issues document, compiled all documentation, created the .msi windows installer, and compiled the project for submittal.

4. Tasks for Next Week

Curtis and Brant must review the project before it is submitted.