Project Plan

Software Engineering Group 6

23/2/2012: Project Plan, v1.0

February 2012 - First Deliverable

Contents: Page no:

Document name: Software Engineering Group 6 / Team Good / Project Plan Doc
Document author: Team Good
Document auditor: Rob Leggatt, Devendra Magar / Team Good Members
Document version: v1.0

23/2/2012: Project Plan, v1.0
Actions: Team Good finalising Project Plan deliverable.

Project Introduction

Project Outline

We have been asked to provide a working implementation of a two-player game. The basis of the game is a simulation of two ant colonies set free in an enclosed 'world'. The world is otherwise populated by sources of food, and rocks which obstruct movement. The ants are tasked with negotiating the terrain of the world to find food and return it to their colony. To help ants navigate and communicate, they are able to mark areas of the world with pheromones which other ants can detect. Ants will also have the ability to kill members of the opposing team by finding and surrounding them. The player whose colony manages to return the most food to their respective base by the end of the simulation wins the game.

Players influence the behaviour of their ants by constructing an 'Ant-Brain' using simple, low-level instructions. Each ant in a given colony uses the same brain, which is compiled before the simulation begins and cannot be changed during the simulation. In this way, the aim of the game is simply to produce the best Ant-Brain.

The main features of the project are:

- A GUI that allows players to upload their Ant-Brains, pick a world, and run the simulation.
- A visualisation of the simulation as it runs, with dynamic feedback on the progress of each player's colony.
- A program to generate randomised worlds to be used in simulations.
- A 'Contest Mode' whereby any number of players may upload Ant-Brains and pit them against each other in a series of simulations to determine an overall winner.
- The design and implementation of a high-level language to facilitate the construction of effective Ant-Brains.

Project Schedule-

Major milestones for the project are as follows:

Deliverable	Due Dates
Project Plan/Group Website	23/02/2012
Requirements Spec./ Acceptance Criteria/ High Level Design Spec.	12/03/2012
Detailed Design Spec.	01/05/2012
Source Code/Test Spec./User Documentation/Peer Assessment	11/06/2012
Presentation to Customer	13/06/2012

Conflict Resolution Plan

Group Conflict

As with all group projects there is the potential for internal conflicts to arise between team members. Any conflicts will be handled by talking things through with the Team Leader. In the event of the team leader being the issue, the leader of the Quality Assurance team will be the port of call. This is to allow people to discuss issues in order to negotiate a resolution that the entire team will be happy with. If it is not possible to come to a satisfactory agreement, the whole group will be brought in to discuss the issue. If the group can come to a consensus (whether by popular vote or otherwise) then the matter will be left at whatever the group decides. If a group member disagrees or still has an issue they may then discuss the issue directly with the tutor. This will also happen if the group are unable to come to a consensus.

With all going to plan any/all problems will have been addressed prior to the end of the course. However, if there is a group belief that a single person has shirked their duties it will be reflected in their peer assessment.

Loss of Group Members

If a group member drops out, an emergency meeting will be held to discuss whether it is still viable to deliver the negotiated extras of the project. If possible people will be reassigned to re-balance the team. Should it be agreed that a particular negotiable or deadline cannot be met in the absence of the team member, the Team Leader will contact the customer to inform them of the problem, and discuss the deliverable.

Deadlines

To prevent a deadline being missed, the following system will be implemented.

- All work will be completed at least 72 hours before a deadline, so the entire team can meet to review the work prior to hand in, eliminating any potential errors in the work.
- The team responsible for the piece of work will complete it at least 24 hours prior to the review to give the team a chance to read through the work prior to the deliverable meeting.
- This will ensure that work is completed at least 96 hours before a deadline so there is still time to make adjustments if they are required.

Note: IF A TEAM IS UNSURE THAT THEY CAN COMPLETE THE WORK IN THE REQUIRED TIME FRAME THEY MUST ALERT THE PROJECT MANAGER AT THE EARLIEST POSSIBLE OPPORTUNITY.

If a deadline is missed due to a team failing to notify the Project manager of a problem, the person(s) responsible will be issued with a formal warning that will be taken into account when it comes to peer assessment. An extension period of 24 hours will be allocated and if possible additional personnel will be assigned to the task.

If the extended deadline is missed, an additional formal warning will be recorded and tutors will be informed who is directly responsible for the missed deadline. If an unofficial deadline is missed then there will not be a penalty unless it causes disruption to the team as a whole. If it causes disruption to another team or delay to the project, the failure will be noted for peer assessment purposes. If the extended deadline (as before, 24 hours) is missed it will be recorded and taken into account when peer assessment comes round.

Work Quality

If a team member does not have expertise in the field that they have selected or have been assigned to for the project, it will be expected that they should seek help either from other team members or tutors before deadlines. If a team submits work which is deemed to be unsatisfactory by the rest of the group a member of the Quality Assurance team will discuss the work with them. They will then be given a chance to improve the work. If the work is still unsatisfactory it will be reflected in the Peer Assessment.

Group Website - teamgood.github.com

The software house will create and maintain a web page which all the Deliverables of the AntWorld project will be available for access. The web page will contain the name of the group (Team Good), software house member names and links to the Deliverables that have been submitted. These documents must always contain the name of the Deliverable and date the link was added in reference to he Configuration Management found in the Quality Manual.

Note, that once the documents have been submitted, modification must NOT occur. However, if modification after submission were necessary, due to detection of problems in the work submitted, then an additional link to a new version of a document shall be allowed.

The Programming team have been assigned the role of creating and maintaining the website, this includes updating it with links to deliverables when they are submitted. The website is scheduled to be completed by 23/02/2012.

Project Phases Plan

The phases of the project will have a structure similar to the Waterfall method. The selection of this prescriptive process model was based on the encountering of a “well-defined project requirements” for the AntWorld Game project. Therefore, there is going to be a sequential approach (one set at a time) in the development stage where it start with customer specification of requirement's (communication) and advances through planning (estimating, scheduling, tracking), modelling (analysis, design), construction (code, test), deployment (delivery, feedback). In order to avoid the disadvantage of this method, such as lack of feedback and tasks dependencies between the sub-teams, the software house intends to use some techniques commonly known as the Agile methods. Some of these practices include: the usage of fair amount of face-to-face communication when necessary and collaboration between team members.

Project Milestones

	Task	A)Internal **Deadline** B) Actual **Deadline**	**Task Duration**	**Dependencies**
A	Project Plan/Group Website	A)Monday 20th Feb 2012 B)Thursday 23 Feb	7 Days	
B	Requirements Spec. Analysis and high-level design	A) Monday 20th Feb 2012 B)Thursday 23rd Feb	14 Days	
C	Acceptance Criteria	A)Monday 30 Apr B)Thursday 3 May	14 Days	B
D	High Level Design Spec.	A)Monday 5 Mar B)Thursday 8 Mar	21 Days	B
E	Detailed Design Spec.	A)Monday 5 Mar B)Thursday 8 Mar	14 Days	B,D
F	Source Code	A)Monday 7 May B)Thursday 10 May	28Days	D,E
G	Test Spec.	A)Monday 7 May B)Thursday 10 May	28Days	F
H	User Documentation	A)Monday 7 May B)Thursday 10 May	28 Days	
I	Peer Assessment	A)Monday 20 Feb B)Monday 23 Feb	14 Days	
J	Customer Presentation	A)Monday 7 May B)Thursday 10 May	3Days	

In the table, the Tasks introduces the assignment(s) for each software house team is to do. Deadlines have two sections, A and B. A is the group finishing date, this is because if any conflicts occur within the process, the group can assign another member to the task, so that the project work meets the customer's expected deadline. B is the official finishing deadline and final submission date. Task Duration specifies the amount of time a team has on a specific deliverable that needs to be completed within the specified time. It can be used to manage organisation for time efficiency. In the table, Dependencies show what deliverables are required for completion so that following tasks may be finished.

Project Milestone (TeamGood)

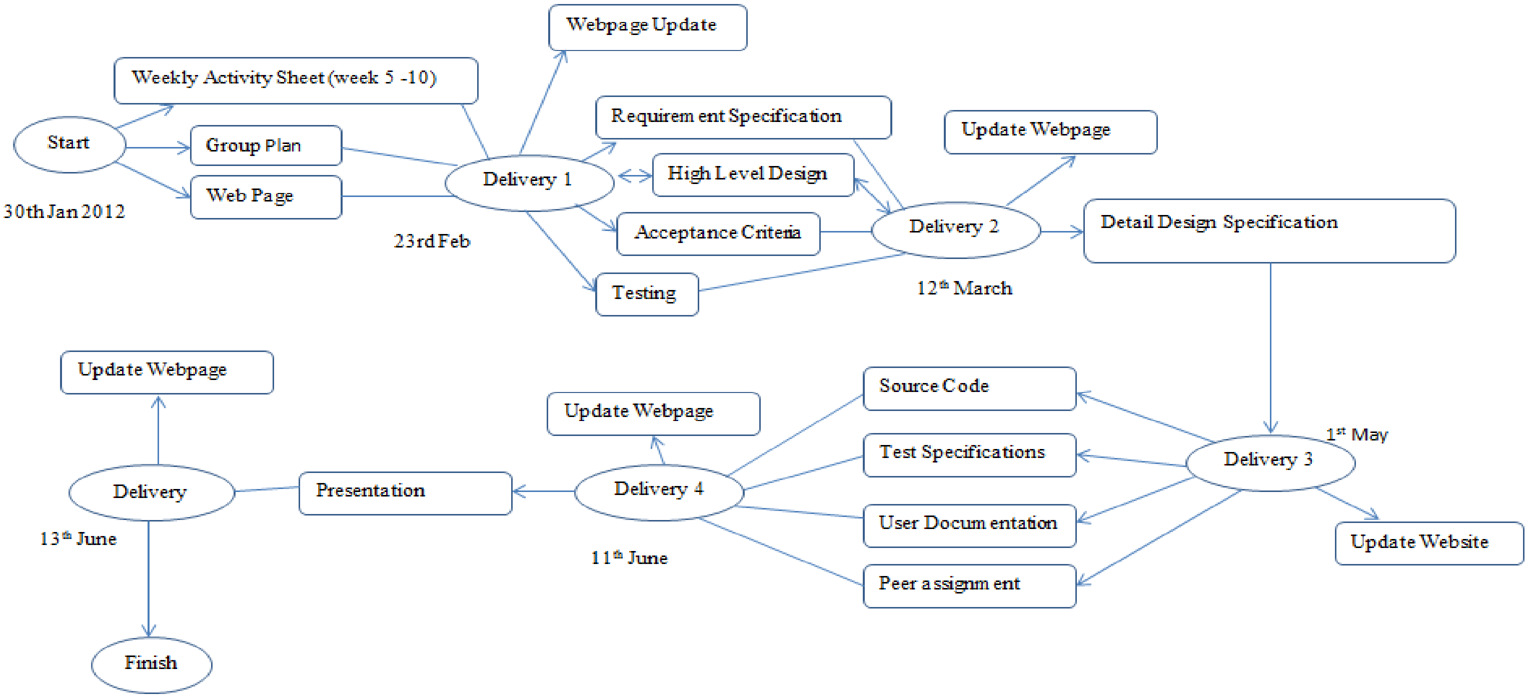


Figure 1. PERT Network for Project Tasks/Deliverable
-------------> Start Line | -------------- isReadyToSubmit | <-----------------> ReferenceTwoDiagram

The Diagram above shows the clear Project Milestones. All team members will follow this diagram to carryout their responsibilities by given time. Below, the detailed project plan shows exactly what each deliverable involves.

Detailed Project Plan:

1

- 1.1 Contents
- 1.2 Introduction
  - 1.2.1 Brief of Project
  - 1.2.2 Base Features
  - 1.2.3 Extra Features
  - 1.2.4 Deadline Timetable

- 1.3 Conflict Resolution Plan
  - 1.3.1 Group Conflict
  - 1.3.2 Loss of Group Members
  - 1.3.3 Deadlines
  - 1.3.4 Quality Issues
- 1.4 Phase Plan
  - 1.4.1 Project Phases
  - 1.4.2 Project Milestones
- 1.5 Organisation Plan
  - 1.5.1 Team Layout
  - 1.5.2 Analysis Team and Responsibilities
  - 1.5.3 Design Team and Responsibilities
  - 1.5.4 Programming Team and Responsibilities
  - 1.5.5 Quality Assurance Team and Responsibilities
- 1.6 Peer Assessment Plan

2 Group Website

- 2.1 Design and Plan : Content/Layout/Style
- 2.2 Implementation of Plan

3 Requirements Spec.

- 3.1 Introduction
- 3.2 Analysis Model
  - 3.2.1 Behaviour
  - 3.2.2 Abstractions
  - 3.2.3 CRC Cards
  - 3.2.4 Scenarios
  - 3.2.5 Class Diagrams
  - 3.2.6 Object Diagrams
  - 3.2.7 Performance Requirements
  - 3.2.8 Constraints of Design

4 Acceptance Criteria

- 4.1 Testing
  - 4.1.1 Determine hardware of test bed
  - 4.1.2 Choose software environment for tests
  - 4.1.3 Determine Normal and Peak load conditions
  - 4.1.4 Determine Necessary data files
  - 4.1.5 Ensure all materials are gathered to allow testing
- 4.2 Acceptance Tests
  - 4.2.1 Finding section to test
  - 4.2.2 Deciding on prerequisites for the test

        - 4.2.3 Describe the test and expected result
- 5 High Level Design Specification
    - 5.1 Introduction
    - 5.2 Architectural Design
        - 5.2.1 Concurrencies
        - 5.2.2 Class Diagrams
        - 5.2.3 Object Diagrams
        - 5.2.4 Class Categorisation
        - 5.2.5 Choose coding style
    - 5.3 Common Tactical Policies
        - 5.3.1 Localised Mechanisms
        - 5.3.2 Handling Policies
    - 5.4 Requirements Cross-Reference
        - 5.4.1 Ensure Corresponding sections have the same names
        - 5.4.2 Provide a reference table for any cross-referencing
- 6 Detailed Design Spec.
    - 6.1 Introduction
    - 6.2 Detailed Design
        - 6.2.1 Identify Abstractions
        - 6.2.2 Object Diagrams
        - 6.2.3 Class Diagrams
        - 6.2.4 Abstract Classes
        - 6.2.5 State Diagrams
        - 6.2.6 Changes from High Level Design Spec
- 7 Source Code
    - 7.1 Coding
        - 7.1.1 Set up version control system
        - 7.1.2 Analyse Design
        - 7.1.3 Allocate responsibilities amongst the team
        - 7.1.4 Complete code and documentation for base simulation
        - 7.1.5 Complete GUI code and documentation
        - 7.1.6 Complete compiler for high-level ant-brain language and integrate into GUI
    - 7.2 Error Checking
        - 7.2.1 Correct errors that become apparent in testing
        - 7.2.2 Re-check through testing
- 8 Test Specification
    - 8.1 Test Scope
        - 8.1.1 Define Functional and Performance Criteria
        - 8.1.2 Define Design Criteria
    - 8.2 Test Plan
        - 8.2.1 Testing Phases
        - 8.2.2 Stub or scaffolding (Overhead software)
    - 8.3 Test Procedures
        - 8.3.1 Test Software description
        - 8.3.2 Overhead software description
        - 8.3.3 Expected results
        - 8.3.4 Test case data
    - 8.4 Test Documentation
        - 8.4.1 Test Results
        - 8.4.2 Corrections that may be required
- 9 User Documentation

9.1 User Documentation
9.2 Installation Instructions
9.3 User Manual
9.3.1 Interface
9.3.2 Functionality
10 Peer Assessment Documentation
10.1 This should be fleshed out for software engineering purposes.
11 Customer Presentation
11.1 Choose Presentation Team
11.2 Presentation Planning
11.3 Presentation

Organisational Plan

Each software team has created their own responsibilities/plans in this section. Each plan has been read through by the team leader and the Quality Assurance team for approval. A Facebook group has been set up to ensure that the team can communicate with all other team members quickly and easily with little hassle. The instant chat function on Google Docs will also be used to allow team members to collaborate while they work. Teams may used other mediums for internal communication that does not concern the rest of the team.

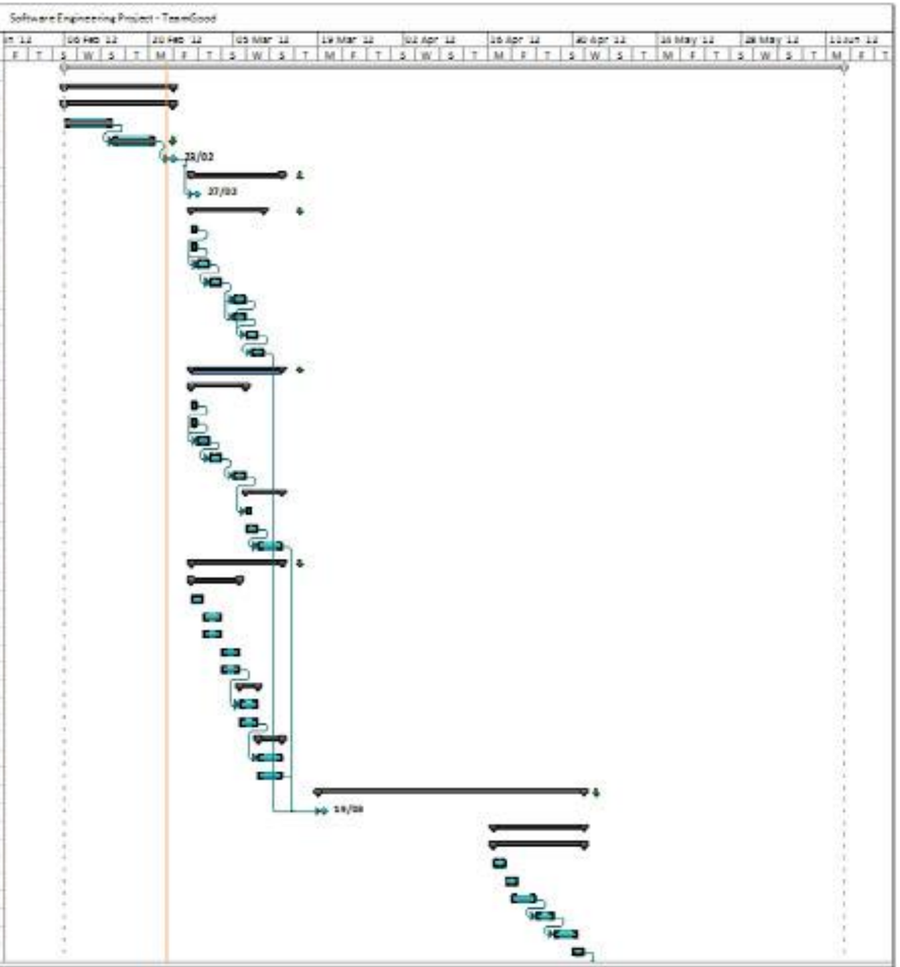
Team Structure: Controlled Decentralised
Team Leader: Rob Leggatt
Design Team: Rob Leggatt, Simon Turner
Analysis Team: Devendra Magar, Wojciech Tolsdorf
Programming Team: David Sheldrick, Josh Pettitt
Quality Assurance Team: Victor Navarro, Wai (Lukaz) Leong

GANTT Chart

A Gantt chart has been produced to represent the flow of the project. The majority of the tasks from the phase plan are represented in the graph, along with estimated time frame for each task. Tasks which were deemed to be to short/insignificant have been removed from the chart to aid simplicity. The Gantt chart has been written around the deliverable deadlines and the internal group deadlines related to each deliverable. However, the group may work faster than this and it may be necessary in future to revise the slack on the project if progress accelerates when compared to initial estimates. For ease of reading, alongside the Gantt chart we have included a calendar that includes all of the same information as the Gantt chart in an easy to read format. (Note: Some tasks did not fit directly on the calendar and have been printed on an overflow page). The calendar shows deadlines more clearly than the Gantt chart as it would have been impractical to print the Gantt chart on a larger piece of paper.

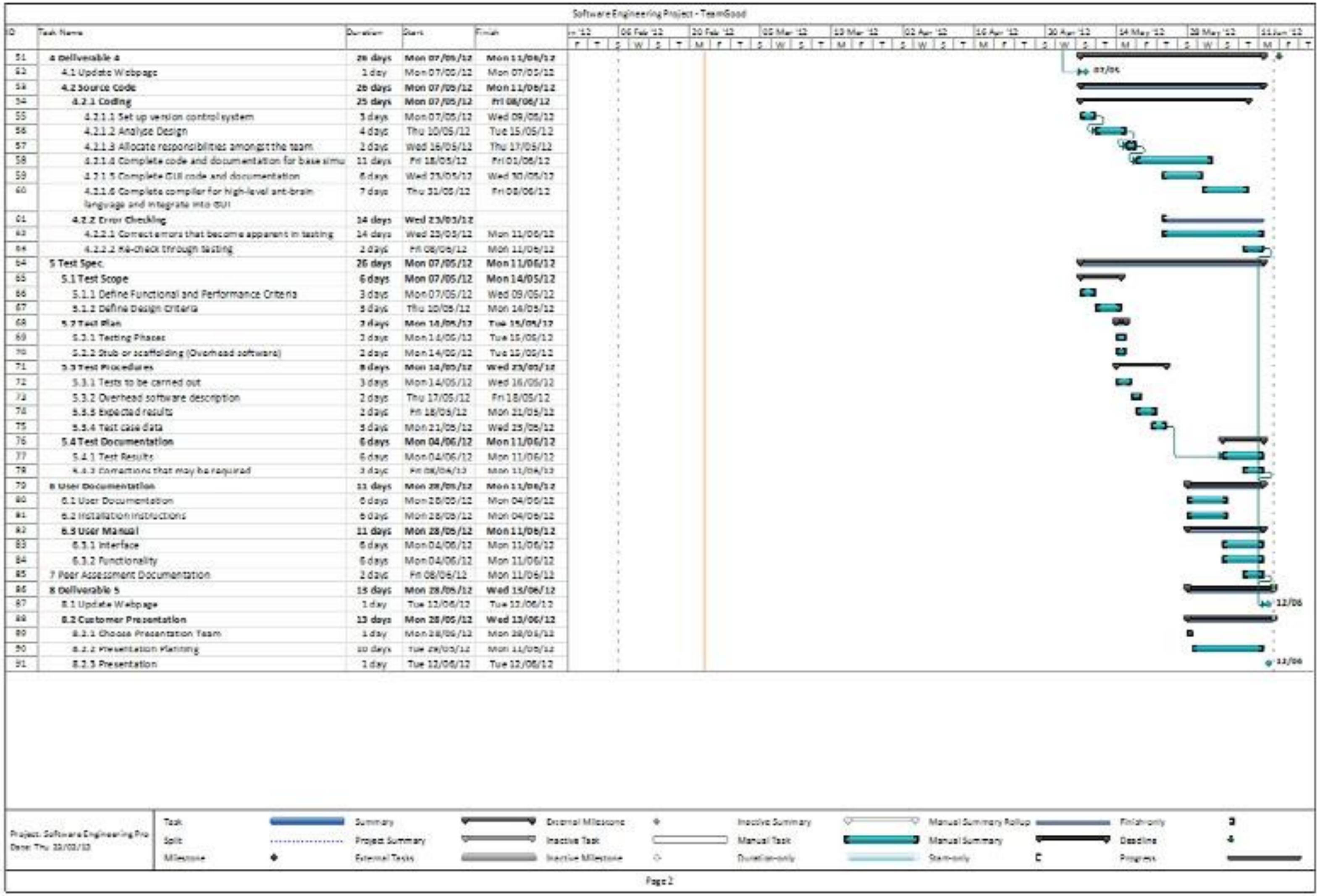
Software Engineering Project - TeamGood

ID	Task Name	Duration	Start	Finish
0	**Software Engineering Project**	**93 days**	**Mon 06/02/1:**	**Wed 13/06/12**
1	**1 Deliverable 1**	**14 days**	**Mon 06/02/12**	**Thu 23/02/12**
2	**1.1 Group Website**	**14 days**	**Mon 06/02/12**	**Thu 23/02/12**
3	1.1.1 Design and Plan : Content/Layout/Style	6 days	Mon 06/02/12	Mon 13/02/12
4	1.1.2 Implementation of Plan	5 days	Tue 14/02/12	Mon 20/02/12
5	1.1.3 Submission	1 day	Thu 23/02/12	Thu 23/02/12
6	**2 Deliverable 2**	**11 days**	**Mon 27/02/12**	**Mon 12/03/12**
7	2.1 Website Update	1 day	Mon 27/02/12	Mon 27/02/12
8	**2.2 Analysis Model**	**10 days**	**Mon 27/02/12**	**Fri 09/03/12**
9	2.2.1 Behaviour	1 day	Mon 27/02/12	Mon 27/02/12
10	2.2.2 Abstraction	1 day	Mon 27/02/12	Mon 27/02/12
11	2.2.3 CRC Cards	2 days	Tue 28/02/12	Wed 29/02/12
12	2.2.4 Scenarios	2 days	Thu 01/03/12	Fri 02/03/12
13	2.2.5 Class Diagram	2 days	Mon 05/03/12	Tue 06/03/12
14	2.2.6 Object Diagrams	2 days	Mon 05/03/12	Tue 06/03/12
15	2.2.7 Performance Reqirements	2 days	Wed 07/03/12	Thu 08/03/12
16	2.2.8 Constraints of Design	2 days	Thu 08/03/12	Fri 09/03/12
17	**2.3 Acceptance Criteria**	**11 days**	**Mon 27/02/12**	**Mon 12/03/12**
18	**2.3.1 Testing**	**7 days**	**Mon 27/02/12**	**Tue 06/03/12**
19	2.3.1.1 Determine hardware of test bed	1 day	Mon 27/02/12	Mon 27/02/12
20	2.3.1.2 Choose software environment for tests	1 day	Mon 27/02/12	Mon 27/02/12
21	2.3.1.3 Determine Normal and Peak load conditions	2 days	Tue 28/02/12	Wed 29/02/12
22	2.3.1.4 Determine Necessary data files	2 days	Thu 01/03/12	Fri 02/03/12
23	2.3.1.5 Ensure all materials are gathered to allow testing	2 days	Mon 05/03/12	Tue 06/03/12
24	**2.3.2 Acceptance tests**	**4 days**	**Wed 07/03/12**	**Mon 12/03/12**
25	2.3.2.1 Finding section to test	1 day	Wed 07/03/12	Wed 07/03/12
26	2.3.2.2 Deciding on prerequisites for the test	2 days	Wed 07/03/12	Thu 08/03/12
27	2.3.2.3 Describe the test and expected result	2 days	Fri 09/03/12	Mon 12/03/12
28	**2.4 High Level Design Specification**	**11 days**	**Mon 27/02/12**	**Mon 12/03/12**
29	**2.4.1 Architectural Design**	**6 days**	**Mon 27/02/12**	**Mon 05/03/12**
30	2.4.1.1 Concurrencies	2 days	Mon 27/02/12	Tue 28/02/12
31	2.4.1.2 Class Diagrams	3 days	Wed 29/02/12	Fri 02/03/12
32	2.4.1.3 Object Diagrams	3 days	Wed 29/02/12	Fri 02/03/12
33	2.4.1.4 Class Categorisation	2 days	Sat 03/03/12	Mon 05/03/12
34	2.4.1.5 Choose coding style	2 days	Sat 03/03/12	Mon 05/03/12
35	**2.4.2 Common Tactical Policies**	**3 days**	**Tue 06/03/12**	**Thu 08/03/12**
36	2.4.2.1 Localised Mechanisms	3 days	Tue 06/03/12	Thu 08/03/12
37	2.4.2.2 Handling Policies	3 days	Tue 06/03/12	Thu 08/03/12
38	**2.4.3 Requirements Cross-Reference**	**2 days**	**Fri 09/03/12**	**Mon 12/03/12**
39	2.4.3.1 Ensure Corresponding sections have the same nai	2 days	Fri 09/03/12	Mon 12/03/12
40	2.4.3.2 Provide a reference table for any cross-referencir	2 days	Fri 09/03/12	Mon 12/03/12
41	**3 Deliverable 3**	**32 days**	**Mon 19/03/12**	**Tue 01/05/12**
42	3.1 Update Webpage	1 day	Mon 19/03/12	Mon 19/03/12
43	**3.2 Detailed Design Spec.**	**11 days**	**Tue 17/04/12**	**Tue 01/05/12**
44	**3.2.1 Detailed Design**	**11 days**	**Tue 17/04/12**	**Tue 01/05/12**
45	3.2.1.1 Identify Abstractions	2 days	Tue 17/04/12	Wed 18/04/12
46	3.2.1.2 Object Diagrams	2 days	Thu 19/04/12	Fri 20/04/12
47	3.2.1.3 Class Diagrams	3 days	Fri 20/04/12	Mon 23/04/12
48	3.2.1.4 Abstract Classes	3 days	Tue 24/04/12	Thu 26/04/12
49	3.2.1.5 State Diagrams	2 days	Fri 27/04/12	Mon 30/04/12
50	3.2.1.6 Changes from High Level Design Spec	2 days	Mon 30/04/12	Tue 01/05/12



Project: Software Engineering Pro
Date: Thu 23/02/12

Task		Summary		External Milestone	◇	Inactive Summary		Manual Summary Rollup		Finish-only	]
Split	…………………	Project Summary		Inactive Task		Manual Task		Manual Summary		Deadline	⇩
Milestone	◆	External Tasks		Inactive Milestone	◇	Duration-only		Start-only	[	Progress	



ID	Task Name	Duration	Start	Finish
51	4 Deliverable 4	26 days	Mon 07/05/12	Mon 11/06/12
52	4.1 Update Webpage	1 day	Mon 07/05/12	Mon 07/05/12
53	4.2 Source Code	26 days	Mon 07/05/12	Mon 11/06/12
54	4.2.1 Coding	25 days	Mon 07/05/12	Fri 08/06/12
55	4.2.1.1 Set up version control system	3 days	Mon 07/05/12	Wed 09/05/12
56	4.2.1.2 Analyse Design	4 days	Thu 10/05/12	Tue 15/05/12
57	4.2.1.3 Allocate responsibilities amongst the team	2 days	Wed 16/05/12	Thu 17/05/12
58	4.2.1.4 Complete code and documentation for base simu	11 days	Fri 18/05/12	Fri 01/06/12
59	4.2.1.5 Complete GUI code and documentation	6 days	Wed 23/05/12	Wed 30/05/12
60	4.2.1.6 Complete compiler for high-level ant-brain language and integrate into GUI	7 days	Thu 31/05/12	Fri 08/06/12
61	4.2.2 Error Checking	14 days	Wed 23/05/12	
62	4.2.2.1 Correct errors that become apparent in testing	14 days	Wed 23/05/12	Mon 11/06/12
63	4.2.2.2 Re-check through testing	2 days	Fri 08/06/12	Mon 11/06/12
64	5 Test Spec.	26 days	Mon 07/05/12	Mon 11/06/12
65	5.1 Test Scope	6 days	Mon 07/05/12	Mon 14/05/12
66	5.1.1 Define Functional and Performance Criteria	3 days	Mon 07/05/12	Wed 09/05/12
67	5.1.2 Define Design Criteria	3 days	Thu 10/05/12	Mon 14/05/12
68	5.2 Test Plan	2 days	Mon 14/05/12	Tue 15/05/12
69	5.2.1 Testing Phases	2 days	Mon 14/05/12	Tue 15/05/12
70	5.2.2 Stub or scaffolding (Overhead software)	2 days	Mon 14/05/12	Tue 15/05/12
71	5.3 Test Procedures	8 days	Mon 14/05/12	Wed 23/05/12
72	5.3.1 Tests to be carried out	3 days	Mon 14/05/12	Wed 16/05/12
73	5.3.2 Overhead software description	2 days	Thu 17/05/12	Fri 18/05/12
74	5.3.3 Expected results	2 days	Fri 18/05/12	Mon 21/05/12
75	5.3.4 Test case data	3 days	Mon 21/05/12	Wed 23/05/12
76	5.4 Test Documentation	6 days	Mon 04/06/12	Mon 11/06/12
77	5.4.1 Test Results	6 days	Mon 04/06/12	Mon 11/06/12
78	5.4.2 Corrections that may be required	2 days	Fri 08/06/12	Mon 11/06/12
79	6 User Documentation	11 days	Mon 28/05/12	Mon 11/06/12
80	6.1 User Documentation	6 days	Mon 28/05/12	Mon 04/06/12
81	6.2 Installation instructions	6 days	Mon 28/05/12	Mon 04/06/12
82	6.3 User Manual	11 days	Mon 28/05/12	Mon 11/06/12
83	6.3.1 Interface	6 days	Mon 04/06/12	Mon 11/06/12
84	6.3.2 Functionality	6 days	Mon 04/06/12	Mon 11/06/12
85	7 Peer Assessment Documentation	2 days	Fri 08/06/12	Mon 11/06/12
86	8 Deliverable 5	15 days	Mon 28/05/12	Wed 13/06/12
87	8.1 Update Webpage	1 day	Tue 12/06/12	Tue 12/06/12
88	8.2 Customer Presentation	13 days	Mon 28/05/12	Wed 13/06/12
89	8.2.1 Choose Presentation Team	1 day	Mon 28/05/12	Mon 28/05/12
90	8.2.2 Presentation Planning	10 days	Tue 29/05/12	Mon 11/06/12
91	8.2.3 Presentation	1 day	Tue 12/06/12	Tue 12/06/12

Project: Software Engineering Pro
Date: Thu 31/05/12

Page 2

February 2012

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
		01	02	03	04	05
06	07	08	09	10	11	12
Software Engineering Project, 93 days Design and Plan : Content/Layout/Style, 6 days						
13	14	15	16	17	18	19
Software Engineering Project, 93 days Design and Plan : Content/Lay	Implementation of Plan, 5 days					
20	21	22	23	24	25	26
Software Engineering Project, 93 days Implementation of Plan, 5 day			Submission			
27	28	29				
Software Engineering Project, 93 days Website Update Behaviour, 1 day Abstraction, 1 day Determine hardware of test b	CRC Cards, 2 days Determine Normal and Peak load conditions, 2 days	Class Diagrams, 3 days Object Diagrams, 3 days				

Overflow Tasks

ID	Name	Start	Finish
20	Choose software environment for tests	Mon 27/02/12	Mon 27/02/12
30	Concurrencies	Mon 27/02/12	Tue 28/02/12

March 2012

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
			01	02	03	04
			Software Engineering Project, 93 days			
			Scenarios, 2 days			
			Determine Necessary data files, 2 days			
		Class Diagrams, 3 days				
		Object Diagrams, 3 days			Class Categorisation, 2 days	
05	06	07	08	09	10	11
			Software Engineering Project, 93 days			
Class Diagram, 2 days		Performance Reqirements, 2 days		Constraints of Design, 2 days		
Object Diagrams, 2 days		Finding section to test, 1 day		Describe the test and expected result, 2 days		
Ensure all materials are gathered to allow testing, 2 days		Deciding on prerequisites for the test, 2 days		Ensure Corresponding sections have the same names, 2 days		
Class Categorisation, 2 days	Localised Mechanisms, 3 days			Provide a reference table for any cross-referencing, 2 days		
12	13	14	15	16	17	18
			Software Engineering Project, 93 days			
Constraints of Design, 2 days						
Describe the test and expecte						
Ensure Corresponding section						
Provide a reference table for a						
19	20	21	22	23	24	25
			Software Engineering Project, 93 days			
Update Webpage						
26	27	28	29	30	31	
			Software Engineering Project, 93 days			

Overflow Tasks

ID	Name	Start	Finish
34	Choose coding style	Sat 03/03/12	Mon 05/03/12
37	Handling Policies	Tue 06/03/12	Thu 08/03/12

April 2012

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
			Software Engineering Project, 93 days			01
02	03	04	05 Software Engineering Project, 93 days	06	07	08
09	10	11	12 Software Engineering Project, 93 days	13	14	15
16	17 Identify Abstractions, 2 days	18	19 Software Engineering Project, 93 days Object Diagrams, 2 days	20 Class Diagrams, 2 days	21	22
23 Class Diagrams, 2 days	24 Abstract Classes, 3 days	25	26 Software Engineering Project, 93 days	27 State Diagrams, 2 days	28	29
30 State Diagrams, 2 days Changes from High Level Design Spec, 2 days			Software Engineering Project, 93 days			

May 2012

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
	01	02	03	04	05	06
Software Engineering Project, 93 days						
Changes from High Level Design Spec, 2 days						
07	08	09	10	11	12	13
Software Engineering Project, 93 days						
Update Webpage						
Set up version control system, 3 days						
		Analyse Design, 4 days				
Define Functional Criteria and Performance Criteria, 3 days						
14	15	16	17	18	19	20
Software Engineering Project, 93 days						
		Complete code and documentation for base simulation, 11 days				
			Test Software description, 2 days			
Analyse Design, 4 days				Expected results, 2 days		
Allocate responsibilities amongst the team, 2 days						
21	22	23	24	25	26	27
Software Engineering Project, 93 days						
Complete code and documentation for base simulation, 11 days						
		Complete GUI code and documentation, 6 days				
Expected results, 2 days		Correct errors that become apparent in testing, 14 days				
Test case data, 3 days						
28	29	30	31			
Software Engineering Project, 93 days						
Complete code and documentation for base simulation, 11 days			Complete compiler for high-level ant-brain language and integrate into GUI, 7 days			
Complete GUI code and documentation, 6 days						
Correct errors that become apparent in testing, 14 days						

Overflow Tasks

ID	Name	Start	Finish
67	Define Design Criteria	Thu 10/05/12	Mon 14/05/12
69	Testing Phases	Mon 14/05/12	Tue 15/05/12
70	Stub or scaffolding?	Mon 14/05/12	Tue 15/05/12
72	Tests to be carried out	Mon 14/05/12	Wed 16/05/12
80	User Documentation	Mon 28/05/12	Mon 04/06/12
81	Installation Instructions	Mon 28/05/12	Mon 04/06/12
89	Choose Presentation Team	Mon 28/05/12	Mon 28/05/12
90	Presentation Planning	Tue 29/05/12	Mon 11/06/12

June 2012

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
				01	02	03
Software Engineering Project, 93 days						
			Complete compiler for high-level ant-brain language and integrate into GUI, 7 days			
Correct errors that become apparent in testing, 14 days						
04	05	06	07	08	09	10
Software Engineering Project, 93 days						
Complete compiler for high-level ant-brain language and integrate into GUI, 7 days						
Correct errors that become apparent in testing, 14 days						
				Re-check through testing, 2 days		
11	12	13	14	15	16	17
Software Engineering Project, 93 days						
		Presentation				
Correct errors that become ap						
Re-check through testing, 2 da						
18	19	20	21	22	23	24
25	26	27	28	29	30	

Overflow Tasks

ID	Name	Start	Finish
80	User Documentation	Mon 28/05/12	Mon 04/06/12
81	Installation Instructions	Mon 28/05/12	Mon 04/06/12
90	Presentation Planning	Tue 29/05/12	Mon 11/06/12
77	Test Results	Mon 04/06/12	Mon 11/06/12
78	Corrections that may be required	Fri 08/06/12	Mon 11/06/12
83	Interface	Mon 04/06/12	Mon 11/06/12
84	Functionality	Mon 04/06/12	Mon 11/06/12
85	Peer Assessment Documentation	Fri 08/06/12	Mon 11/06/12
87	Update Webpage	Mon 11/06/12	Mon 11/06/12

Project File Space(s):

- Teamgood.github.com
- teamgoodg6@gmail.com
- Google Docs

Responsibilities:

Team Leader Responsibilities:

1. Will be the communication hub between the team.
2. Each team will report to the team leader for any problems and queries.
3. Be the medium of communication between the customer and the team.
4. Will ensure Team meetings are regular.

Team Chair Responsibilities:

1. Overseeing the production of deliverables
2. Arranging and chairing Group Team meetings and Deliverable Review meetings

Team Recorder Responsibilities (Quality Assurance Team)

1. Agendas and Minutes for Group Team meetings,Deliverable Review and Customer meetings
2. Circulating Agendas and Minutes

Analysis Team Plan & Responsibilities (Devendra Mgr, Wojciech Tolsdorf):

The Analysis team is responsible for the production of the Analysis Model for the Software Engineering project, by working closely with the Design Team. The team plan is to focus on the construction of the Requirements Specification document and the analysis of the system as a whole by producing appropriate UML diagrams for the required software, which can later be presented to the customer.

The UML diagrams together with the Use Case diagrams will define how the potential users will interact with the program. Once the customer requirements are ready and approved, the team will begin creating scenarios of user interaction and user goals. Those goals and scenarios will be important for the Quality Assurance team during the construction of the Test Specification. They will also be helpful for the Design Team when making decisions about the User Interface Design. In addition to the UML Diagrams, the Class-Relationship-Collaboration (CRC) cards will also be produced to plan what classes are necessary for the project and how they will interact with each other.

As mentioned above, the Analysis Team will collaborate with the Design Team during the production of High-Level Design to make sure that it fully meets the Customer Requirements. The Customer Requirements will be assessed based on pre/post conditions, source of input, destinations of output and side effects.

Finally, the Analysis Team will be also responsible for the production of the User Documentation. The User Documentation will include the User Licence, User Manual and the Installation Procedure. These will be produced in collaboration with the Design and Programing teams, as they will have detailed insight on the proposed system. Also, if necessary, the Analysis Team will be assisting the Design and Programming teams in the design of the graphical user interface to make sure that it meets the Customer Requirements.

Individual Responsibilities:

- Team Representative(Devendra Magar)
- Recorder(Wojciech Tolsdorf)

Project phase and individuals responsibilities:

- Project Plan - Devendra Magar & Wojciech Tolsdorf
- Requirement Specifications - Devendra Magar & Wojciech Tolsdorf
- User Documentation - Devendra Magar & Wojciech Tolsdorf

Elements of the analysis model and staff allocation:

Flow-oriented Elements

- Data flow diagrams

Staff allocated to complete the task – Devendra
Schedule completion – Spring - Week 8/Thursday

Scenario-based Elements

- Use-cases-text
- Use-case diagrams
- Swim lane diagrams
- Activity Diagrams

Staff allocated to complete the task – Devendra Magar and Wojciech Tolsdorf
Schedule completion – spring – Week 9/Thursday

Elements of Analysis Model
Behavioural Elements

- State diagrams
- Sequence

Staff allocated to complete the task – Wojciech Tolsdorf

Schedule completion – spring - Week 10/Thursday

Class-based elements

- Class Diagram
- Analysis packages
- CRC models
- Collaboration diagrams

Staff allocated to complete the task – Devendra Magar
Schedule completion – Spring - Week /10/Thursday

PERT CHART
Project Milestones – Analysis Team

Deliverable	
Start week	**Duration**
Group Deadline	**Submission Deadline**

Project Plan	
Week 4	14d
30/01/2012	13/02/12

Data Flow	
Week 4	2d
24/02/2012	14/03/12

Swin lane	
Week 4	2d
24/02/2012	14/03/12

Use cases	
Week 7	2d
20/02/2012	22/02/12

Activity	
Week 7	2d
20/02/2012	22/02/12

Collaboration	
Week 8	3d
27/02/2012	1/03/12

CRC	
Week 8	3d
27/02/2012	1/03/12

Class	
Week 8	3d
27/03/2012	1/03/12

Sequence	
Week 7	2d
03/03/2012	14/03/12

State	
Week 7	2d
03/03/2012	14/03/12

Design and Programming Phases

User Documentation	
Week 7	4d
28/5/2012	31/5/2012

Summer – Week 8
Spring – Week 10
Spring – Week7
* Use Case includes (Use-case text and Use-case diagram)
* Class diagram includes (Class diagram and Analysis Packages)

Design Team Plan & Responsibilities (Rob Leggatt, Simon Turner):

Design Team Responsibilities

The Design team will be responsible for organising the classes that the Analysis team have identified. They will organise how the classes will interact and work with each other, as well as producing an appropriate layout for the program in UML. Working as a bridge between the Analysis and the Programming team. Having collaborated with both the Analysis and the Programming team, they will decide the coding style for the project. Later on, they will produce a more in depth Design document to assist the Programming team in producing the best interpretation of the original Analysis Model.

Detailed Design Specification

- Introduction - Rob
- Architectural Design -
  - Concurrences - Simon
  - Class Diagrams - Simon
  - Object Diagrams - Simon
  - Class Categorisation - Rob
  - Coding Style - Rob
- Common Tactical Policies -
  - Localised Mechanisms - Simon
  - Handling Policies - Simon
- Requirements Cross Referencing
  - Ensuring Corresponding Sections have the same names - Rob
  - Provide a reference table for cross referencing - Rob

- Detailed Design Spec.
  - Introduction - Rob
  - Object Diagrams - Simon
  - Class Diagrams - Simon
  - Abstract Classes - Simon
  - State Diagrams - Rob
  - Changes from High Level Design Spec - Rob

PERT CHART

Project Milestones – Design Team

Deliverable	
Start week	**Duration**
Group Deadline	**Submission Deadline**

Project Plan	
Week 4	24d
30/01/2012	23/02/12

Detailed Design Spec	
Week 1	11d
17/04/12	30/04/12

High Level Design Spec	
Week 8	11d
27/02/12	09/03/12

Collaboration	
Week 3	80d
21/02/12	08/06/12

Summer – Week 8
Spring – Week 10
Spring – Week7

Programming Team Plan & Responsibilities (David Sheldrick, Joshua Pettitt):

The Programming team will have three main areas of responsibility:

- Collaboration with the Design team to facilitate the production of a workable and well-structured program architecture.
- Source code implementation of the program.
- Collaboration with the Quality Assurance team to carry out effective testing and bug-fixing.

The first task is to set up version control management. This will be done using Git, hosted on github.com. An appropriate branching model will be decided upon.

Before coding can begin, the Detailed Design specification will be mapped onto a series of programming tasks which will then be broken down into appropriate sub-tasks and allocated to team members. An iterative testing plan will be devised to run alongside code development. This process of assigning tasks and coding/testing will be done three times sequentially for the following program components:

1. Base simulation with parsers
2. GUI
3. High-level ant-brain language compiler

Individual Responsibilities:

Team Representative: David Sheldrick
Team Recorder: (David Sheldrick, Joshua Pettitt)

Project phase(s)and individuals responsibilities:

- Collaboration with the Design Team to produce a robust Program Architecture (David Sheldrick, Joshua Pettitt)
- Project Plan - (David Sheldrick, Joshua Pettitt)
    - Organisation Plan - (David Sheldrick)
    - Phase Plan - (David Sheldrick, Joshua Pettitt)
- Implementation of Source Code - (David Sheldrick)
- Troubleshooting and Bug - fixing - (David Sheldrick, Joshua Pettitt)
- Implementation of website and online repository - (David Sheldrick, Joshua Pettitt)

PERT CHART

Project Milestones – Programming Team

Deliverable	
Start week	**Duration**
Group Deadline	**Submission Deadline**

Project Plan	
Week 4	24d
30/01/2012	23/02/12

Group Web Site	
Week 7	17d
06/02/2012	23/02/12

Source Code	
Week 3	26d
07/05/12	31/05/12

Error Checking	
Week 5	14d
23/05/12	08/06/12

Documentation	
Week 8	7d
04/06/12	11/06/12

Testing Collaboration	
Week 3	80d
21/02/12	08/06/12

Quality Assurance Plan &Responsibilities (Victor Navarro, Wai (Lukaz)

Leong):

The Quality Assurance Team has five core responsibilities that consist of:

- Production of their respective parts of the Project Plan; their sections of the Phase Plan and the Organisation Plan
- Production of the document deliverable: Acceptance Criteria
- Production of the document deliverable: Test Specification
- Adherence to the guide line protocols specified in the Quality Manual i.e. Configuration Management, production of Activity Sheets
- Production and recording of the Agendas and Minutes for all meetings especially Deliverable Review Meetings

All of the five primary responsibilities stated above are the core founding tasks that the Quality Assurance Team must ensure that are properly attended to. The team must thoroughly complete all necessary deliverable documents and conduct and ensure an obligatory standard based on the Configuration Management and Quality Manual is satisfied for the entire Group.

The Quality Assurance Team also have secondary responsibilities that include an overall look out for the productivity of the entire Group. As well as that, they must cater for the meetings providing all the necessary means of producing and circulating Agendas and Minutes.

Moreover, the Quality Assurance Team are an overseer for the Team Leader in the context of helping if conflicts arise and if further problems occur based on deliverables as well.

Acceptance Criteria:

The Acceptance Criteria is a document deliverable that is to be created by the Quality Assurance Team where it will contain the description of designed and chosen tests that are to be carried out on the designed and developed software program for the game. It is a document that will ensure that it meets the Requirements Specification.

There are two parts to be completed for the Acceptance Criteria:

- **Test Environment**

This section will have the description of the environments that will be used to carry out the tests for the software. It will also specify what kind of machine is used as well as that, multiple platforms of software that has been anticipated to be able to display and allow interactivity for the user audience. It also includes the load conditions and the data files necessary for the tests to be conducted.

- **Acceptance Tests**

Based on the Analysis Model that has been developed by the Analysis Team, the Acceptance Tests will be structured/based off of that model so that each test will have the following information provided (in respect to the specifications in the Analysis Model), Prerequisites (data files) to allow the test(s) to run and the test performance and its expected results recorded.

The Acceptance Tests is a 'black box test'. A development of a 'white box test' will be found in the Test Specification, the secondary document deliverable of the Quality Assurance Team.

Test Specification:

The Test Specification will hold the descriptions of how all modules have been produced during the design and programming/implementation of the source code for the software. Each of these modules/phases will be tested accordingly within the Test Specification.

Constituent parts of the Test Specification:

- **Scope**

The scope is the place where the description of the functionality, performance and design criteria for the software that is to be tested, can be found. It will be dependent on the Specification requirements as reference to it as well the Design hierarchy will be required. The scope is where a description of how the tests are appropriate for the schedule described within the Project Plan.

- **Test Plan**

This is where the 'black box' and 'white box' testing will be found. In reference of the Acceptance Criteria deliverable, this section of the Test Specification will have the testing activities into phases that will have descriptions of the tasks for the software.

It will include:

- Test Phases: A list of the testing phases and the tasks involved for each.
- Overhead software: For each phase, the stub or scaffolding it requires will be listed.

- **Test Procedures**

Within the Test Procedures a series of documentation is to take place; where four elements of the Test Procedures can be found. They are all relevant to the point where a test can be differentiated and identified, this is the key for the development of the test to better and enhance the overall quality of the software and it's testing.

The Test Procedures will have:

- **Test Descriptions:** Describing the tests carried out for specific elements of the software and other content.
- **Overhead software description:** A description of the utilised scaffolding or stub used within the phase(s).
- **Expected Results:** The resultant outcomes of the tests conducted, a characterisation of a successful test will be stated here.
- **Test Case Data:** A collection of any data files that will be relevant for use to conduct the test will be documented here.

- **Test Results**

As there will be phases within the constituent parts of the Test Specification, the Test Results will serve as the place of archiving of the results of running each of the test phases. Of course, these phases can be iterated due to the expectancy of failed tests, therefore a double check ensures quality assurance within the development of not only the software but also the entirety of the project.

All of the tests will be documented and each test must have the following recorded accordingly:

- **Results**: A bold statement about the produced results.
- **Status**: A statement on whether the test was successful or not.
- **Action**: A statement that precludes that because a test was unsuccessful, a corrective act should be carried out. This includes an assigned team member to carry out the test as well as who should perform to take the necessary action(s) for the test.

All testing will use the JUnit testing framework found within the local machines.

Individual Responsibilities:

- Team Representative: Victor Navarro
- Recorder(s): Victor Navarro & Wai (Lukaz) Leong

Project phase(s)and individuals responsibilities:

- Project Plan - Victor Navarro & Wai (Lukaz) Leong
  - Organisation Plan - Victor Navarro
  - Phase Plan - Waik (Lukaz) Leong

- Acceptance Criteria - Victor Navarro & Wai (Lukaz) Leong

  - Test Environment - Victor Navarro
  - Acceptance Tests - Wai (Lukaz) Leong

- Test Specification - Victor Navarro & Wai (Lukaz) Leong
  - Scope - Victor Navarro
  - Test Plan - Wai (Lukaz) Leong & Victor Navarro
    - Test Phases - Wai (Lukaz) Leong
    - Overhead Software - Wai (Lukaz) Leong & Victor Navarro
  - Test Procedures - Wai (Lukaz) Leong & Victor Navarro
    - Test Descriptions - Victor Navarro
    - Overhead Software description - Wai (Lukaz) Leong
    - Expected Results - Wai (Lukaz) Leong & Victor Navarro
    - Test Case Data - Wai (Lukaz) Leong & Victor Navarro

- Test Results - Wai (Lukaz) Leong & Victor Navarro
  - Results - Victor Navarro
  - Status - Wai (Lukaz) Leong
  - Action - Wai (Lukaz) Leong & Victor Navarro

- Management of adherence to the Quality Manual (Configuration Management) - Victor Navarro & Wai (Lukaz) Leong
  - Activity Sheets - Wai (Lukaz) Leong & Victor Navarro
  - Reviewing of Documents - Wai (Lukaz) Leong & Victor Navarro
    - Configuration Management - Wai (Lukaz) Leong & Victor Navarro
      - Preliminary - Victor Navarro
      - Checked in - Wai (Lukaz) Leong
      - Checked out - Wai (Lukaz) Leong & Victor Navarro

- Production of Agendas and Minutes - Wai (Lukaz) Leong & Victor Navarro

Both members of the Quality Assurance team will alternate starting from Victor Navarro, for example the first two meetings will be documented and recorded by Victor and the next two by Lukaz. These will be posted on the forums available for viewing for all members of the group with efficiency and accessibility in mind. Moreover, the documents will also be available in the file spaces of the group.

Overview of Phase Plan for Quality Assurance: Test Specification

In order to provide a good final program to the customers, processes testing and debugging is required. At this stage, we will test the program with a series of tests to see if the program has any bugs that is preventing it from being able to run smoothly and correctly. The program will be ensured that the bugs are fixed by the Quality Assurance team and programming team if there are any bugs appeared on the program. If there is no problem with the program, the Quality Assurance team will make sure that the program is up to standard before delivering to the customers. The main tasks that the Quality Assurance team will cover are listed below:

- **Smoke testing**

This is just a quick review of the program. This is a test that would be done soon as the program is completed by the programming team, so the Quality Assurance team could take more testing in depth as shown below:

- **Black Box testing**

Black box testing is one of the most basic testing. This black box testing is not basely making test on the logical structure of the program, it is mainly making test on the user interface and the functional of the program. The Quality Assurance team will make test for the following options:

- Is the user interface good enough for the customers?
- Is input data and output data correctly working with the program?
- Does the program compile correctly with the customer requirements specification?

- **White Box testing**

White box testing is a more advance testing, this also meaning that this does require more programming skills. This white box testing is to mainly to work with the code, to check the structure of the code, logic of the code, find the problems of the code and debug it if there are any problems.

- **User Acceptance testing**

If possible, the Quality Assurance team would like to invite the customer to test the program to see if they are happy with the current program. We could possibly deliver some changes if the customer is not quite happy in some aspects.

From the testing tasks above, we could make sure that the program is up to standard and it is what the customer required. More importantly, the program will be running correctly at any points. During the testing, the Quality Assurance team may have to work with the other sub teams including Programming, Analysis and Design team to debug/ improve the program before handing the program to the customer.

PERT CHART

Project Milestones – Quality Assurance Team

Deliverable	
Start week	**Duration**
Group Deadline	**Submission Deadline**

Project Plan	
Week 4	24d
30/01/2012	23/02/12

Acceptance Criteria	
Week 10	11d
27/02/2012	09/03/12

Test specification	
Week 7	111d
21/02/2012	11/06/12

Quality Manual Protocol	
Week 5	107d
07/02/12	24/05/12

Meeting Recording	
Week 3	91d
23/01/12	24/05/12

Collaboration	

Week 3	91d
23/01/12	24/05/12

Overseeing Team	
Week 3	91d
23/01/12	24/05/12

Testing Collaboration	
Week 3	80d
21/02/12	08/06/12

Summer – Week 8
Spring – Week 10
Spring – Week7

Peer Assessment Plan

For the Peer Assessment, there will be multiple opportunities for group members to rate each other. In order to implement this in a fair way, the following system has been devised:

Each person is given 100 points to split between the other members of the group. Each member will distribute their points between the rest of the members of the team, according to the amount of effort they believe that the person has put in on the specific deliverable as a whole. If they believe that a person has put in the amount of effort required of them for the deliverable they should give around 14 points as this is the average. If they believe a person has done more/ less work than was required of them they will award more/less points as necessary. Only whole values of points can be distributed.

Example:

Person 1 has 100 points to give to persons 2-8.

Person 1 gives Person 2 13 points as they feel Person 2 has done what is required of them.
Person 1 gives Person 3 0 points as Person 3 has not attended meetings or done any work.
Person 1 gives Person 4 20 points as they have done slightly more work than is required.
Person 1 gives Person 5 8 points as they have done slightly less work than they were required to do.
Person 1 gives Person 6 30 points as they have done far more work than they were required to, even helping other teams.
Person 1 gives Person 7 16 points as they feel Person 7 has done more than is required of them, though not as much as Person 4.
Person 1 gives Person 8 13 points as they feel Person 8 has done what is required of them.

This can then be tabulated to give a group consensus on whether a person has reached what should be 100% of the work required of them. This will mean that if everyone has done the work required of them they should all receive around 100 marks. This record will be made for EVERY deliverable, so as to give people who have under performed a chance to put more effort in later to redeem themselves. The results of every deliverable will not be submitted for marking, however the final results taken at the last meeting in the summer term will be.

On top of this percentage system of peer review, it will be noted if a team member is deemed to have produced work that is either of notable quality, be it good or bad. A log of these events will be submitted alongside the percentile review at the end of the course. The final mark out of 100 received that will be submitted for marking will be divided by 8 in order to conform with the specification for peer review. If there are any decimal values for marks then they will be distributed automatically (.1/.2/.3/.4 of a mark will be knocked off, .6/.7/.8/.9 of a mark will be rounded up). In the event of two people receiving .5 of a mark, the person with less marks will be rounded up. In the event of two people receiving .5 of a mark and both having the same number of marks, it will be up to the team to decide who receives the extra point (decided by popular vote). If this proves to be a stalemate it will noted in the documentation that they share the final mark.

Marks will be tabulated by the Team Leader, and the Team Leader will keep the ratings that people submit confidential so as to prevent team disputes. If any team member develops a problem with the Team Leader managing marks, they are free to raise it at any time. In the event of this occurring it is likely that management of the marks will be handed to a tutor, who will pass the final results back to the Team Leader.

Bibliography

There is no official course textbook we shall be using, because there are no right or wrong most books are so, we will pick from different books and lecture slides. Some of the book we shall use.

- Summerville, Software Engineering, 9th edition.
- R. S. Pressman, Software Engineering: A Practitioner's Approach, 6th edition. Another good resource for software engineering, if somewhat verbose.

Subsidiary texts are:

- Stevens, P. and Pooley, R., Using UML, Updated ed, Addison-Wesley, 2000. There may be newer editions by now.
- Gamma, E., Helm, R., Johnson, R. and Vlissides, J., Design Patterns, Addison-Wesley, 1995. This book is written in pseudo-C++ notation which should be easily readable for competent Java programmers. The web is full of translations of the design patterns in this book to Java (and other languages).