Team 9 – Project Plan

Version 1.0

Authors: Arsalan Sadeghpour, Dan Read, Jeremiad Oluwakanmi, Kea Tossavainen, Regan Ware, Sam Marsh

Content:

1. Requirements
    a. Domain
    b. Functional
    c. Non-Functional
2. Project Plan
    a. Planning / documentation
    b. Design
    c. Detailed Design Specification
    d. Coding / Implementation
    e. Ant Brain
    f. Testing
3. Staff Organisation
    a. Roles
    b. Communication, change and something management
    c. Analysis team responsibilities
    d. Design team responsibilities
    e. Programming team responsibilities
    f. Quality Assurance team responsibilities
4. Task Times
5. Perth Chart

1. Requirements

The team shall create an ant game and an ant brain which will take part in a competition against other teams in their ant worlds.

a. Domain requirements

There are no additional domain requirements for this project. The application domain of the system is assumed to be extremely similar to the development and testing domain: as per the non-functional product requirements, this software is designed to run on personal operating systems compatible with Java 8.

All current requirements have been classified as functional or non-functional.

b. Functional requirements:

The program shall check if ant brain is valid
The program shall check if ant world is valid
The program shall visualise ant world
The program shall allow two players to play
The program shall keep statistics and determine the winner of the game based on those
The program shall allow tournaments when more than two players submit an ant brain and players will be paired to play against each other

- **Ant brain:**

Ant brain shall be a finite state machine
Ant brain shall be able to sense their surroundings (Here, ahead, LeftAhead, RightAhead)
Ant brain shall be able to move to ahead, LeftAhead and RightAhead cells with a 14 turn rest after

- **Ant:**

Each ant shall have unique id that determines the order in which ants take actions
Each ant shall have an integer between 0 and 9999 which represents current state of brain
Each ant shall carry at most one particle of food, this shall be represented as boolean field
Each ant shall have a "colour" denoting team
Each ant shall have a "resting" integer which represents the number of turns they must stay inactive after moving
Each ant shall have a has_food boolean recording whether the ant is carrying food
If an ant is surrounded by 5 or more enemy ants it shall detect it and remove itself
If an ant is cornered but only surrounded by 4 ants of the other species, it shall not die
When an ant dies it shall drop 3 food onto the tile it was on
Each ant shall keep track of their current direction
Ant shall have several different actions:

- Sense surroundings
- Mark or unmark a cell
- Pick up food from a current cell
- Drop food
- Turn left or right to change direction
- Move to an adjacent cell
- Flip

- **Ant world:**

Ant world shall be hexagonal grid
Each cell in the grid shall either be rocky or clear, or contain an ant hill that is red or black
Ant hills shall not be adjacent to each other
Ants should not go to each others ant hills
Each clear cell in the grid can contain at most one of each of these:

- At most one ant
- Non-negative number of food between 0 and 9
- At most one set of chemical markers from 6 different markers

- **UI Requirements:**

Shall display the world including tile contents
Scales with world dimensions (x,y)
Ideally a swing GUI interface
Can be text based due to time constraints (Print to IDE output)

c. Non-functional requirements:

- **Reliability**

The software should crash no more than 1 time in 1000 when parsing user files, even if the files are corrupt or not well-formed.

The software should crash no more than 1 time in 1000 of the time when simulating a game or tournament.

The GUI should be stable and responsive to user input at least 99% of the time, including when a game is running. This applies only to systems with more than 400MB of free RAM.

The software shall parse user files of up to 10,000 lines each without crashing more than 1 time in 1000.

Visualisations of games and tournaments should proceed with no visible defects (glitches) at least 99 times out of 100.

If the program crashes, a log containing the stack trace of the error and any other pertinent details shall be written to the program's working directory, titled "ant-game-crash-[current unix time].log".

- **Usability**

The software should have a minimal learning curve for new users. That is, the GUI should have intuitive and clear controls that step the user(s) through each stage of the game without being required to follow external instructions (not including understanding the rules of the game and the development of a user's ant-brain file).

All user interfaces shall be developed using the JavaFX API.

The software shall present a help menu, accessible at all times in the program, which upon clicking displays a popup user interface containing the searchable user documentation for the program.

- **Performance**

Response time of the system when interacting with any component of the GUI should be negligible before the user is presented with the interaction's result or with a loading screen.

The GUI should not hang (that is, be unresponsive to user interaction) for more than one second after any user interaction.

Parsing user files (ant-brain, ant-world) should take no longer than 5 seconds.

The two-player game simulation should take no longer than 30 seconds for 300,000 rounds, not including artificially introduced delays when displaying the game to the user.

- **Security**

The software shall not require internet or network access to run, and will have access to the full feature-set of the program offline without use of any network connections.

The system shall not purposefully access or alter any existing user or system files, with the exception of ant-brain and ant-world files chosen with explicit permission from the user (via selection from a file-chooser interface).

- **Portability**

The software shall be cross-platform across commonly used personal desktop operating systems (Windows 7+, Mac OS X 10.8.3+, Linux) that support Java 8.0.

- **Resources**

The system should not store more than 100MB of non-volatile data, not including the user's ant-brain and ant-world files.

The software shall use no more than 100MB of RAM, with the exception of a maximum of 400MB of RAM while executing/displaying a game.

- **Deployment**

A build of the program shall be delivered in executable JAR format, to allow the user to run the program without accessing the command line and without any other external configuration by the user.

- **Organisation**

All development and testing shall be undertaken in either the Netbeans IDE or IntelliJ IDE.

Team communication relevant to the product-related aspects of the project (planning, development, testing) shall be recorded for future reference:

Facebook chat - message archive

Slack chat - message archive

In-person - meeting minutes

JUnit 4.12 shall be used as the testing library across all development and testing systems.

Gradle 2.5+ shall be used as the build automaton system across all development and testing systems.

All project-related documents, source code, tests and builds shall be kept on the team's GitHub repository, with new versions being committed and pushed to the repository on a regular basis and as soon as a major change/refactoring occurs.

The software's source code shall be documented extensively, defined as follows:

The Javadoc comment structure shall be used to document every field, method and class.

The structure of Javadoc documentation shall follow the Oracle style guide.

Single-line comments shall be used inside methods to make the purpose of lines/blocks of code clear, when deemed appropriate by the coder(s).

For fragments of incomplete code which are pushed to the project's GitHub repository or left to complete later, a single line comment starting with "//TODO: " shall be used to briefly describe the task to complete.

A copy of all project documents and source code shall be stored independently on each team member's personal computer, regularly pulled from the central team GitHub repository. In addition, an up-to-date backup of the GitHub repository shall be kept in one or more of the team's personal Dropbox repositories.

The Jacoco code coverage analysis tool shall be used by the testing/validation team to ensure maximum code coverage. The final submitted system shall have 100% code coverage.

- **External**

The software shall be delivered by Thursday 5 May at 4:00PM, by online submission to Study Direct.

All libraries and external material used in the project shall be documented and credited to the original author(s), as per the University of Sussex's plagiarism policies.

2. Project Plan

a. Planning/Documentation

This phase of the project will involve the writing up and compilation of multiple different documents, plans and reviews that will detail relevant information and specifications relating to the project team; the ‘client’; and the project itself. It is also fair to assume some documentation may need to be completed or be updated over the course of the project but the majority should be produced at this stage. The main documents to be produced at this stage should include the following:

b. Design

The design phase of the project will involve drawing up, analysing and deciding upon the design of the Ant Game, within the confines of the requirements given in the Requirements Document. As a result, this part of the project will lead to useful diagrams (e.g. UML Diagrams, Flowcharts etc.) and other documents being produced at this stage such as:

c. Detailed Design Specification

High-Level Design Specification The design phase will be very important for trying to implement a successful Ant Game and at least understanding the fundamentals of how our team will go about making the game and the different aspects of it (e.g. Ant Brain, World Viewer, GUI, etc.).

d. Coding (Implementation)

This phase of the project will involve writing up the code for and implementation of the Ant Game, based on the specifications given from the Design phase and the Requirements Document. The elements to be coded for and implemented are as follows:

e. Ant Brain

Ant Brain Parser

Ant World Parser

Ant World Viewer

Random Ant World Generator

Two Player Mode

Tournament Mode This stage in the project will have the members of the team working in part, both individually and together on programming the code for the mentioned elements, where some easier elements to be coded for can be done individually and the harder elements/parts to code for can be worked on in pairs/subgroups.

f. Testing

The Testing Phase of the project will most likely run simultaneously alongside coding; to make the coding process run more smoothly and to identify problems and errors early, preventing potentially more difficult complications (allowing time to be saved that would otherwise be spent looking for errors) down the line. Of course this phase will have to be run according to a:

Test Specification that will highlight how the tests should work.

3. Staff Organisation

a. Roles

The team consists of Sam, Kea, Jeremiah, Regan and Arsalan. The Project Manager and Technical Leader will be Sam and he will be responsible for critical technical decisions, however the team structure is democractic therefore decisions are formulated as a team. Kea, Dan and Jeremiah will take part in Analysis formulating ideas with the Design team composed of Arsalan and Dan who will communicate with the programming team composed of Regan and Sam to discuss feasible design aspects. Finally Jeremiah and Arsalan will be part of quality assurance working with Regan and Sam to discuss solutions to problems. The PERT Chart and the management of documentation will be managed by Kea, Jeremiah and Sam who will work the rest of the team to ensure the project is on track.

b. Communication, Configuration and Change Mangement

The main communication channel will be Slack where discussions will take place, whereas for information sharing of technical documents Github will be used as it includes version control procedures making it easier to make changes. The team will each store a Project Log keeping track of documents they upload and edit, each document will consist of it's name, author, auditor, version number and last modification date and when a document is uploaded and edited it is communicated across in Slack and reviewed at the next meeting before being approved as further changes may be required. If changes are required this is communicated in Slack or in a team meeting and is later implemented.

c. Analysis Team Responsibilities

The overall responsibilities of Analysis will be to read the project brief in detail and work on the requirements engineering document which will denote of the required functionality of the system and it's constraints. The next step is to communicate this across to the design team who will read the specification and use it as an aid to the design process. The final step is to make any required changes/clarifications to the design team if a requirement is not fully understood.

d. Design Team Responsibilities

The overall responsibilities of Design will be to meet the software specification outlined by the analysis team. The team will establish an overall architecture for the system by

implementing a high level and detailed design by providing appropriate design documentation in UML visually representing the abstractions identifed by the analysis team and will formulate these into relationships which are later used by the programming team as aids to the implementation of the project. The next step is to refactor the design for the case of the programming team identifying design flaws.

e. Programming Team Responsibilities

The overall responsibilities of Programming will be to implement the design effectively using the supplied documentation, whilst also communicating with the design team any potential flaws in the design. The programming team will also work with quality assurance by helping to supply an effective test plan. The programming team will also have an ongoing of step of refactoring any code for the case of design changes communicated by the design team.

f. Quality Assurance Team Responsibilities

Quality Assurance will ensure the programming team's implemented solution is thoroughly tested against the requirements identified by the analysis team and communicate any errors which fail to meet these requirements up to the programming team who will make the required changes if necessary. QA will first implement a test plan whilst working with the programming team and design team and then will physically conduct various forms of testing.

4. Task Times

This document gives an approximate optimistic-realistic-pessimistic estimation for the time spent on each task in the project. All times are given in days.

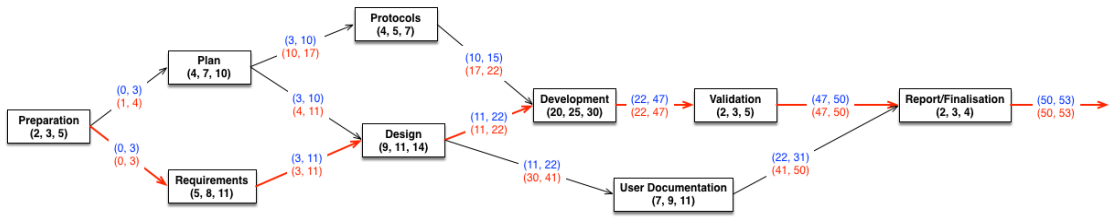
Task	Optimistic	Realistic	Pessimistic
Preparation			
Team Organisation	1	2	3
Configuration Management	2	3	5
Plan			

Task	Optimistic	Realistic	Pessimistic
Conflict Resolution	1	2	3
Phase Plan	4	7	10
Organisation Plan	3	5	7
Peer Assessment Plan	1	2	3
Protocols			
Acceptance Criteria	2	3	4
Development Protocols	1	2	3
Test Specification	4	5	7
Requirements			
Functional	5	8	11
Non-Functional	3	6	9
Domain	1	2	3
Design			
High-Level	4	6	9
Low-Level	9	11	14
Development			

Task	Optimistic	Realistic	Pessimistic
Programming	20	25	30
Documentation	5	7	9
Unit Testing	10	12	14
Validation			
Component Testing	2	3	5
Finalisation			
User Documentation	7	9	11
Report	2	3	4

5. PERT Chart

The PERT chart below shows the main phases of the project. It is assumed that the subtasks of each phase (as in the above table) can be worked on asynchronously by the team, so the longest sub-task for each phase has been used to characterise the total completion time of each phase. In the chart the (x, y, z) tuple below each node is the optimistic, realistic and pessimistic completion time in days. The realistic completion time has been used to calculate the critical path (the path marked with red arrows).



The project was started by the group on 18/02/2016. This means, assuming each phase on the critical path is completed in exactly the above 'realistic' number of days, the completion

date of the project should be 11/04/2016. This gives TODO days of freedom in the critical path before the project deadline.