CS 461 - Fall 2016 - Requirements Document

Project DevAI

Jacob Broderick, Kristen Patterson, Brandon Chatham

Abstract

This project is to create an agent to play the game Starcraft Brood War, a real time strategy game created by Blizzard Entertainment. After this project is complete, it could be expanded upon by future students in a club setting. This project will be a template so that future students can strive to develop better solutions than those developed within this project. This document will provide the details of the project as well as provide a list of tasks the development team must complete. The goal of this document is to specify the requirements for the system to be developed.

CONTENTS

I. Introduction

This section covers the purpose and the overview of this document. It also includes a list of abbreviations and definitions as well as describes the scope of the project. Any sources referenced throughout the document are also included.

A. Purpose

The purpose of this document is to describe the requirements of ”Project DevAI”. This document will also describe the purpose of the project as well as name some of the constraints that will be involved in the process of development. The intended audience for this document is the client and the development team. The client can make sure that their specific concerns and constraints have been met, while the development team can use this document as a reference during the development process.

B. Scope

”Project DevAI” is an AI that plays Starcraft Brood War and provides a user with a well-documented AI that can be used for research purposes. The AI should implement at least one strategy to play Starcraft Brood War. A user can interact with the AI by opening the source code and viewing it. The source code must be highly documented and modular that way users can pull any section of code from the source code and use it for their own program.

C. Definitions, acronyms, and abbreviations

AI - Artificial intelligence

Agent - The program that will play Starcraft Brood War by itself

Client - Person or organization that has requested the project to be developed

User - Any future Oregon State University club member that wishes to interact with the system

Starcraft Brood War - A real time strategy game created by Blizzard Entertainment

API - Application Programming Interface

BWAPI - API used to interact with Brood War’s code

Script - Procedure in a C program that calls a variety of generic real time strategy procedures

Micro - A term used in Starcraft to describe actions pertaining to controlling individual units, mostly related to attacking

Macro - A term used in Starcraft to describe actions pertaining to managing resources and constructing buildings, mostly related to economy management

Unit - A controllable object in Starcraft that can move, attack, and gather resources

Resource - An item in Starcraft used to construct units and buildings

D. References

[1] IEEE Software Engineering Standards Committee, IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, October 20, 1998.

E. Overview

This document contains two more sections. The second section contains information about the product perspective and functions. It also details the user characteristics and discusses the various constraints and dependencies the program has. The third section provides the specific and functional requirements of the system.

II. Overall description

A. Product perspective

This system consists of a single application that will be ran via the game Starcraft Brood War. The application will communicate with Starcraft via an API that will inject itself directly into the game code. The software has no external interfaces with which it can be modified in real time, it must be completely autonomous and run as long as the game itself is running. The agent is allowed to save files to a folder that will be removed once the game is done. There are no constraints to how much memory is accepted by the system, but it must be able to handle the data without slowing down the frame rate of the game. The agent can create a log of what errors occur as it is running so that developers have the required information to fix any bugs. It will be limited to the operations defined in the Brood War API that will be used to create the project.

B. User interfaces

There are two kinds of users that could interact with the machine, developers and players. There shall be a way for the system to provide log files so developers can see the results of individual runs of the game. Users can play against the the agent as well, but it would require no changes from normal use. All interaction occurs within the game parameters. Players would be able to try different strategies against the AI to see how it reacts to different situations. Other AIs could also be players in our system. They would be pit against our AI to play against each other. There will be no other interaction with the users, as the AI should be able to work completely autonomously.

C. Constraints

The system must be able to run within Starcraft Brood War on most modern computers. It needs to be able to operate within the memory constraints to not affect the runtime of the environment. The system must not slow down the game by more than 1 frame per 10 seconds.

D. Apportioning of requirements

Other strategies are not included in the initial plan. If the system is completed early, then other strategies will be implemented. The AI's ability to change strategies will be delayed until future versions. The initial version shall only have one script that can be executed efficiently. The project will be opened to future developers to add more features to in a club setting. These are undocumented for now, but may be written in the future. The library will be separated into classes focused on the two generalized areas of game-play categories: micro and macro.

III. Specific Requirements

The agent should be comprised of modular components that are well-documented. Functionality should be broken into stand-alone pieces that could nearly directly translate into another developer's agent. Also, these pieces should be documented well enough such that a developer can understand and make adjustments to them per their needs. Our stretch goal is to give the agent the ability to switch between scripts based on game circumstances. This would mean adjusting to a new strategy based on the agent's resources and potentially the opponents estimated resources.

A. Functional requirements

Our agent will use the BWAPI to interact with the Starcraft Brood War game. Our agent will follow a scripted strategy. Since the intent is for the agent to play the game on its own, there is no user interface beyond running the program. People can potentially play against the agent, however this requires no additional interaction with our agent. The scripted strategies will be based on how we can engineer the agent to gather resources and expend them while also eliminating the opponent's ability to do these things.

1) Micro library objects: Objects focused on micro management decisions will often be more specific to the current state of the game and micro management decisions of the opponent.

a) Forming groups: An object for unit positioning will encapsulate the functionality of unit grouping. Units are generally grouped for combat when facing an opponent and this object will focus on ensuring combat units are moving in a group as much as possible based on input from the combat decisions object.

b) Movement: This is an object for individual unit movements and group movements. Our base strategy will focus primarily on moving aggressive units into advantageous positions for attack and defense and doing so in groups. Other strategies that may be developed in following versions may be more heavily focused on individual unit movements for specific strategies. These actions will be made based on input from the combat decisions object.

c) Attacking: This object will handle the commands of aggressive tactical decisions. The combat decisions object will extensively interact with this object in order to attack enemy units or structures.

2) Macro library objects: Objects focused on macro management will be more closely tied to basic functionality of the game rather than tactics while keeping track of the resource collection and expenditures.

a) Resource gathering: This object will handle resource collection while working closely with the state of the game in order to keep track of what resources will be needed most immediately.

b) Building production: As resources are gathered, the building production object will establish a foundation for production of necessary units for the scripted strategy. This object will work closely with the construction decision object in order to produce necessary buildings as soon as the resources are available.

c) Unit production: The unit production object will focus on production of units when resources are made available based on input from the unit production decision making object.

d) Expansions: Expansions will come about through an interaction of all three of these Macro subsections' decision object. A new gathering location will be established, new units will be built to populate the resource deposits, and resources will be gathered.

3) Decision making objects: Each object requires decision making logic to decide what to do under certain circumstances. The decision making objects will decipher what input to give the objects given a set of circumstances.

a) Combat decisions: This object will handle decisions based on combat information such as opponent positioning or unit types. Also, tactical objectives must be considered, for instance, whether a unit is defending or attacking a specific area.

b) Resource gathering decisions: Resource gathering requires decision making when deciding whether to harvest from the home base resource deposits or expansion resource deposits while also considering opponent positions. Sometimes, resource deposits are at maximum capacity of resource collecting units and therefore, the unit should collect from a less populated expansion base if one exists.

c) Construction decisions: Structures are allocated a specific amount of area when being built and cannot overlap. This object will handle the decision making with regards to where structures should be built.

d) Unit production decisions: Units will be produced based on the scripted strategy, available resources, and sometimes, opponent unit production. In later versions, this logic will more extensively respond to opponent actions. For now, this object will follow the scripted strategy very closely and execute commands for units that fit the strategy.

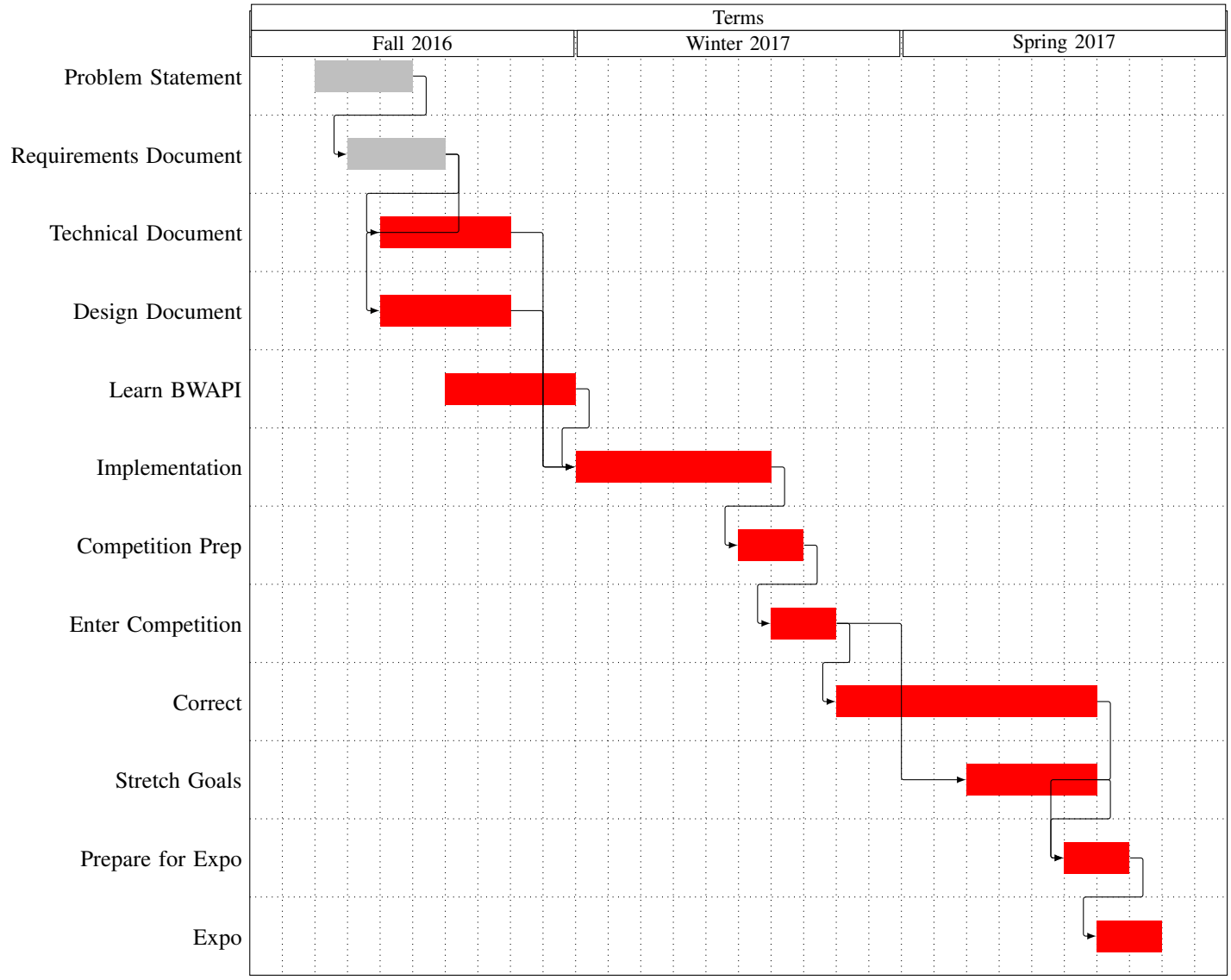

Fig. 1. Gantt Chart