**TIGER ZONE**

**Group: E**

Software Design and Development Documentation

CEN3031 – Software Engineering

Instructor: Dave Small

07 November 2016

# **TABLE OF CONTENTS**

CONTENTS

[TABLE OF CONTENTS 2](#_Toc466243452)

[PROJECT PLANNING 4](#_Toc466243453)

[OVERVIEW AND PRODUCT VISION STATEMENT 4](#_Toc466243454)

[PROJECT PLAN 5](#_Toc466243455)

[TEAM BREAKDOWN 5](#_Toc466243456)

[COMMUNICATION 6](#_Toc466243457)

[UPDATES 7](#_Toc466243458)

[RELEASE PLAN & WORK BREAKDOWN STRUCTURE 7](#_Toc466243459)

[REQUIREMENTS AND SPECIFICATION 9](#_Toc466243460)

[USER NEEDS & PRODUCT BACKLOG 9](#_Toc466243461)

[ARCHITECTURE AND DESIGN 11](#_Toc466243462)

[SYSTEM DESIGN 12](#_Toc466243463)

[USAGE DIAGRAM 13](#_Toc466243464)

[SUB-SYSTEM DESIGN 13](#_Toc466243465)

[ENTITY SYSTEM 14](#_Toc466243466)

[BOARD SYSTEM 15](#_Toc466243467)

[SCORING SYSTEM 16](#_Toc466243468)

[ACTION SYSTEM 17](#_Toc466243469)

[ARTIFICIAL INTELLIGENCE SYSTEM 19](#_Toc466243470)

[NETWORK SYSTEM 20](#_Toc466243471)

[TESTING AND IMPLEMENTATION 22](#_Toc466243472)

[UNIT TESTING 22](#_Toc466243473)

[ENTITIES 22](#_Toc466243474)

[BOARD SYSTEM 23](#_Toc466243475)

[SCORING SYSTEM 23](#_Toc466243476)

[ACTION SYSTEM 23](#_Toc466243477)

[ARTIFICIAL INTELLIGENCE SYSTEM 24](#_Toc466243478)

[NETWORK SYSTEM 24](#_Toc466243479)

[ACCEPTANCE TESTING 24](#_Toc466243480)

[APPENDICES 25](#_Toc466243481)

[APPENDIX A: CARCASSONNE GAME RULES 25](#_Toc466243482)

# **PROJECT PLANNING**

This section of the design documentation serves to communicate the overview of the project, the current proposal of solving the problem, and the current plan on how to go about solving the problem. The entire document is a preliminary report on what is currently believed to be the client’s intention, and is likely to change during the remainder of the term.

## **OVERVIEW AND PRODUCT VISION STATEMENT**

**In this sub-section, a general overview of the project’s problem, its goal, and its scope will be provided in the form of a high-level description. It is intended to serve as a high-level view of our product vision statement without explicitly demonstrating so.**

Carcassonne is a popular tile-based, turn-style board game that is named after the medieval fortified town of Carcassonne in southern France. Tiger Zone is the product that has been given to us by the client and will be what this project will be focusing on. As far as we can tell at this point, Tiger Zone will simply be a rendition of Carcassonne with modifications that will arise later and will not be in line with the original board game.

Be that as it may, Tiger Zone is a term project with project specifications provided by the Software Engineering course instructor—Dave Small. The intent of this project is to simulate a real-world situation where the product owner has requested a non-specific product to be created, where design requirements and details will be updated as time moves along. We are to implement agile design tactics to account for this, as well as a clean architectural approach.

It is in this simulation where we must then incorporate SCRUM practices, use sagacious architectural design development, and produce system, design, and testing artifacts to best demonstrate the team’s approach. The end goal of this is to have developed a working implementation of Tiger Zone that is line with the client’s requirements. These said requirements are described, in detail, in the documentation that follows.

## **PROJECT PLAN**

This subsection will serve as a layout for the structure of our team, the resources we will be using, and the general process that will be adhered to in order to best tackle the problem at hand.

### **TEAM BREAKDOWN**

Due to the nature of academia, it is difficult to mimic scrum practices with actual groups due to the variability in everyone’s schedules and obligations. Despite this, an attempt to adhere to it as much as possible will be made, trying best to keep the group as organized. A scrum master was designated to facilitate the agile process as much as possible; moreover, everyone is still considered to be a part of the development team and expected to make contributions to the design itself. Below is a mapping of the team members and which subsystem they have been assigned to focus on (Table 1). In this way, that team member will become an expert on that subsystem and be the focal point everyone goes to for understanding when it comes time to implement. This team member will also be largely responsible for creating any testing artifacts deemed necessary.

|  |  |
| --- | --- |
| **TEAM MEMBER** | **SUBSYSTEM DESIGNATION** |
| Matthew Booe – **scrum master** | Entity |
| Connor Ward | Board |
| Tana Konda | Scoring |
| Zachary Mills | Artificial Intelligence |
| Charley Chau | Action |
| Josiah Crepeau | Entity & Network |

Table 1: Team Member Distributions

Despite this, every team member is expected to make contributions to all other subsystems as much as possible to ensure a timely release schedule. Pair programming will also become a part of our process, in that every piece of code committed to by a team member will be reviewed by at least one other team member so that there is fluid understanding and consistency in the design amongst the rest of the team. In any group meeting, any individual can take on the role of a meeting note taker as it will be difficult for all group members to meet at one time. The purpose of the note taker is to take down any meeting notes to distribute to the team so that everyone is caught up with the current design.

### **COMMUNICATION**

To streamline our design and development process, there was a undoubtable need to have various modes of communication and control between the project group members. Due to the nature of the course in an academic setting, it is particularly difficult to manage and ensure that everyone is on the same page as everybody else as we do not work in the same environment at all times. Thus, the following mediums have been chosen to better the flow of collaboration and control among the group:

* **GroupMe** – a group messaging application designed for multiple platforms
  + Used for distributing information to everyone that deems immediate concern
* **Google Drive** – a file storage and synchronization service
  + Used for distributing and maintaining any documents or deliverables that are relevant to the project
* **GitHub** – a web-based Git repository hosting service
  + Used for version control, access control, task management, and source code management of working software
  + Handles a working product backlog of features that are necessary to implement for the end product
  + Current working repository: <https://github.com/Mirdaki/tiger-zone/>

### **UPDATES**

The project plan is an artifact of our design process that is intended to imitate agile practices as best as possible. Because of this, the project plan is subject to change incrementally as the design process persists and may or may not be the same by project end. All working copies of this document and backlogs will be under direct version control handled by both GitHub and Google Drive, and will be automatically reflected in the most recent version.

Any and all updates to the source code itself will also be handled by GitHub where a corresponding commit message will be viewable for every change made. These changes will adhere to both the design that we compose in the initial process and any pattern or necessary feature that was overlooked or added by the product owner.

### **RELEASE PLAN & WORK BREAKDOWN STRUCTURE**

This sub-subsection is a *very* high level depiction of the release plan and work breakdown structure as it is currently seen. Since this design process will be in accordance with scrum practices, each working software release will be modeled in the form of sprints. As such, the following time table shows the expected objective of what should be completed before the next sprint begins in the form of deliverables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SPRINT** | **#1** | **#2** | **#3** | **#4** |
| **RELEASE** | 7 NOVEMBER 2016 | 14 NOVEMBER 2016 | 21 NOVEMBER 2016 | 28 NOVEMBER 2016 |
| **OBJECTIVE** | Requirements deciphered.  System design, development design, and testing artifacts completed and/or analyzed. | Have board, tile, entity, scoring, and state systems in place. Conduct tests. | Have AI and interaction systems completed. Conduct tests. | Have game completed and in working condition, able to communicate over network. Conduct tests. |

Table 2: High Level Project Release Plan and Deliverables

The following table is to serve as a step process for the team’s current work breakdown structure in the form of steps and what sprint each step is mapped with. Again, because this is design process in line with agile practices, much of the work breakdown structure and above deliverables are subject to change as necessary.

|  |  |  |
| --- | --- | --- |
| **STEP** | **DESCRIPTION** | **SPRINT** |
| 1 | **Project Overview given by Product Owner** | 1 |
| 2 | **Requirements** | 1 |
| 2.1 | Requirements gathering | 1 |
| 2.2 | Requirements specification | 1 |
| 2.3 | Requirements validation | 1 |
| 3 | **Elaboration** | 1 |
| 3.1 | High-level design | 1 |
| 3.2 | High-level design breakdown | 1 |
| 3.3 | Low-level object design | 1 |
| 3.4 | Elaboration review | 1 |
| 4 | **System construction and implementation** | 2 |
| 4.1 | Entity system | 2 |
| 4.1.1 | *Testing code implementation* | *2* |
| 4.2 | Board system | 2 |
| 4.2.1 | *Testing code implementation* | 2 |
| 4.3 | Scoring system | 2 |
| 4.3.1 | *Testing code implementation* | 2 |
| 4.4 | Action system | 3 |
| 4.4.1 | *Testing code implementation* | 3 |
| 4.5 | Artificial intelligence system | 3 |
| 4.5.1 | *Testing code implementation* | 3 |
| 4.6 | Network system | 4 |
| 4.6.1 | *Testing code implementation* | 4 |
| 4.7 | Subsystem integration | 4 |
| 5 | **Acceptance Implementation Testing** | 4 |
| 5.1 | Test planning | 4 |
| 5.2 | Test code implementation | 4 |
| 6 | **Documentation** | 4 |
| 7 | **Reflection** | 4 |
| 7.1 | Post-mortem report and group evaluation | 4 |

Table 3: Project Step Process

# **REQUIREMENTS AND SPECIFICATION**

This section will serve as a detailed description of what the ending product must demonstrate. The requirements and specifications are laid out firstly by the product owner, which are then analyzed as much as possible by the team in the form of user needs, use cases, user stories, etc. The following is a concise bulleted list of product specifications that are immediately given as overall constraints to the problem and are agreed upon by the development team and project stakeholders:

* The product owner wants the development team to create a game called Tiger Zone
* The game will be a rendition of the popular board game known as Carcassonne
* Project specifications are subject to change, and the rendition will not be a complete copy of the game Carcassonne
* The rules of Tiger Zone can initially be assumed to follow the same basic rules of Carcassonne (**see Appendix A**)
* Tiger Zone will be able to support up to two players maximum
* Tiger Zone will be able to handle artificial intelligence for some, or all, players involved
* Tiger Zone must be able to communicate over a network
  + This network will be implemented over TCP
* Any illegal moves by a player will be considered a forfeit
* If a player runs out of their allotted time, it will be considered a forfeit
* The player with the most points is declared the winner
* The design for this game must clearly demonstrate “clean” architectural design, in that most framework decisions and considerations are no more than “details”

## **USER NEEDS & PRODUCT BACKLOG**

The user needs we uncovered will be listed in the form of a product backlog, as follows. Each entry is given an ID, description, and estimate on the amount of time it might take to implement. Furthermore, each entry is listed in decreasing priority. As usual, all contingencies in this list are subject to change and may alter as time goes on—especially in terms of priority, estimation, and what needs to be achieved.

|  |  |  |
| --- | --- | --- |
| **ID** | **ENTRY** | **ESTIMATE (h)** |
| 1 | As a player, I want to be able to start a new game. | 3 |
| 2 | As a player, I want to be able to choose to play against another person or a computer. | 3 |
| 3 | As a player, I want to be able to see available and remaining tiles. | 4 |
| 4 | As a player, I want to be able to see the current drawn tile from the tile bag. | 2 |
| 5 | As a player, I want to be able to know where other tiles are located. | 3 |
| 6 | As a player, I want to be able to know available placement locations based on the currently drawn tile. | 5 |
| 7 | As a player, I want to be able to rotate a tile. | 2 |
| 8 | As a player, I want to be able to place down a tile. | 3 |
| 9 | As a player, I want to be able to place down a meeple on either a road, city, field, or cloister. | 6 |
| 10 | As a player, I want to be able to confirm or deny my decision and backtrack if needed. | 2 |
| 11 | As a player, I want to be able to see my score and my opponents score at all times. | 12 |
| 12 | As a player, I want to be able to see my potential score increase upon tile placement. | 5 |
| 13 | As a player, I want to know how many meeples I and my opponent have available. | 3 |
| 14 | As a player, I want to be able to see where meeple are currently placed on the board. | 3 |
| 15 | As a player, I want to be able to know if the move I am about to do is valid or not. | 7 |
| 16 | As a player, I want to know when I can pick up my meeple. | 6 |
| 17 | As a player, I want to be able to know how much allotted time I have in the game. | 2 |
| 18 | As a player, I want to be able to connect with other players over a network as a player or as an AI. | 12 |
| 19 | As a player, I want to be able to play two games simultaneously. | 6 |
| 20 | As a player, I want to know when a game ends | 1 |
| 21 | As a player, I want to be able to know my score at the end of the game | 3 |
| 22 | As a player, I want to have the option to be able to play again | 2 |

Table 4: Product Backlog

# **ARCHITECTURE AND DESIGN**

This section will serve as an overview of the current architectural design as it stands. Its intention is to model a hexagonal architecture with as many design patterns as possible, so that when changes do arise any modifications the development team must administer will be minimal at best.

To best emulate hexagonal and “clean” architecture, the development team has made the decision to model the design by combining boundary-control-entity design practices with hexagonal design practices, as illustrated in the diagram below.

C:\Users\Josiah\AppData\Local\Microsoft\Windows\INetCacheContent.Word\download.png

Figure 1: Hexagonal + ECB Design

In this style, actors interact with entities (i.e. the application game itself, Tiger Zone) directly through boundary objects. These boundary objects will then administer instructrions to the controller objects, who then work as a medium between boundaries and entities. To relate this to hexagonal architecture, actors are our “external agencies”, boundary objects are our “ports and adaptors” that communicate with the controllers, and entities comprise our “application”. Every layer is separate from one another, and no outer layer should know anything about the inner layers.

In particular, the following table illustrates our actors, boundaries, controllers, and entities as per the requirements analysis. This analysis is preliminary, and as development ensues, changes are sure to arise.

|  |  |  |  |
| --- | --- | --- | --- |
| **Actors (External Entities)** | **Boundary Objects** | **Controllers** | **Entities** |
| * TCP Server * Players * Computer AI | * FIT * Junit * Net * GUI * CLI | * Scoring handler * Interaction handler * Tile handler * Board handler * AI handler * Network handler | * Players * Tiles * Board * Meeple * Terrain * Tile stack |

Table 5: ECB + Hexagonal Components

## **SYSTEM DESIGN**

The following is a systematic overview of the components that will potentially make up Tiger Zone as a whole. Each subsystem was identified according to the needs of the program, and were consolidated into the

* Entity system
* Board system
* Scoring system
* Action system
* Artificial intelligence system
* Network system

Each will be accompanied by a brief description, collaborators, UML diagrams, and anything else that may help to get the intention across and was uncovered during the elaboration phase of the development process. These are also subject to change as the approach taken may very well be vastly different than what we currently comprehend.

### **USAGE DIAGRAM**

This is a usage activity diagram to help illustrate general process as its currently understood in using the application being developed. It is high-level in that it does not depict any specifics that are to be made in the decision making nor does it indicate any restrictions or constraints on a given activity.

C:\Users\Josiah\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Untitled Diagram (1).png

## **SUB-SYSTEM DESIGN**

The following subsections will outline each individual subsystem that was aforementioned above. Each system will have a brief description, which other systems it collaborates with, any necessary UML diagrams, and whatever else is needed to effectively describe what the system is trying to achieve. These are sure to change as better understanding of the problems come about.

### **ENTITY SYSTEM**

***Description Overview***

The entity system is designed to be responsible for handling any and all situations involving entities within Tiger Zone. The entities involved were described in the preceding section, and are mostly independent of one another. Although they themselves are independent, all other systems make use of these entities in one way or another and are integral in ensuring the system works as a whole.

***Collaborators***

All other systems.

***UML***

C:\Users\Josiah\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Entities (10).png

Figure 2: Entity System UML Diagram

### **BOARD SYSTEM**

***Description Overview***

The board system is in charge of handling all responsibilities regarding the game board itself. It manages all running connected regions, used and unused locations, meeple placements, tile placement validation, active players, and scoring. Due to its nature, all other systems are incorporated into the board system in one way or another.

***Collaborators***

All other systems.

***UML***

C:\Users\Josiah\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Board System (2).png

Figure 3: Board System UML Diagram

### **SCORING SYSTEM**

***Description Overview***

The scoring system is a concurrent system in that any time a tile is placed, regions are connected, meeples are placed, or the game ends, it is called to score the current running board. It is mainly involved with the board and action systems. Due to the nature of the system, it is best represented with the following UML activity diagram.

***Collaborators***

Board, Action

***UML***

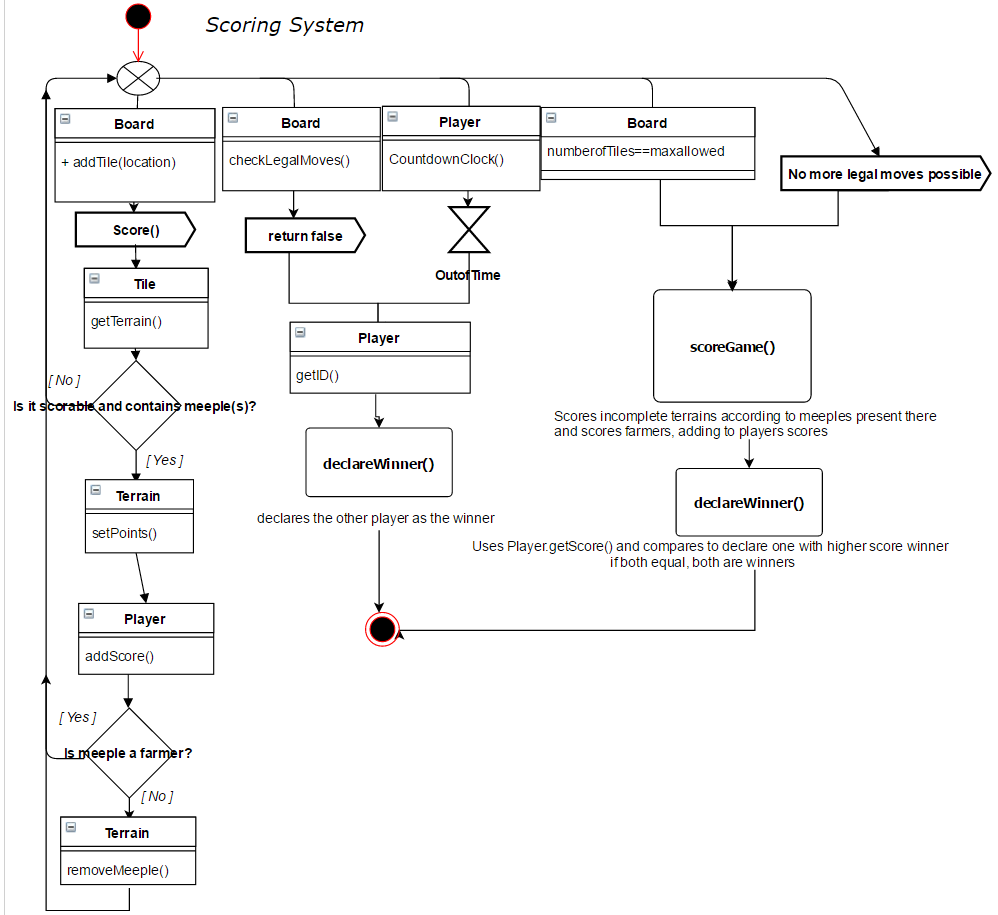


Figure 4: Scoring System UML Activity Diagram

### **ACTION SYSTEM**

***Description Overview***

The action system is the interface handler that drives the remainder of the systems. Any sort of action within the game is handled directly by the action system that serves to call other systems as needed. Due to its nature, it is best represented as an activity diagram to illustrate the concurrent processes that should occur one after another.

***Collaborators***

All other systems.

***UML***

C:\Users\Josiah\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Action System.png

Figure 5: Action System UML Activity Diagram

### **ARTIFICIAL INTELLIGENCE SYSTEM**

***Description Overview***

The responsibility of this handler is to manage all of the attributes and methods to be used by the artificial intelligence algorithm that will be implemented. As of current, the general approach is known and is difficult to express in terms of an actual UML diagram. However, what follows is a depiction of said approach to the best of our current understanding of the issue.

***Collaborators***

All other systems.

***UML***

C:\Users\Josiah\AppData\Local\Microsoft\Windows\INetCacheContent.Word\AI Diagram.png

Figure 6: Artificial Intelligence UML Logic

### **NETWORK SYSTEM**

***Description Overview***

Not much is currently known about the networking system other than that the server communication will be handled over TCP. Be that as it may, further research into TCP seems to indicate that an object needed to be passed over a server must be serializable into a bytestream for sending and receiving. A serialized object is merely putting the object into an understandable format that can be decoded by either party. The serialized object is then passed via socket and handled by either the server or the client. The project specification seems to indicate that the last placed tile, player information, score, and game status messages are all likely objects subject to be sent. Further information on network implementation will reference http://www.cs.mcgill.ca/~adenau/teaching/cs303/lecture12.pdf

***Collaborators***

Entity, Board

***UML***

C:\Users\Josiah\AppData\Local\Microsoft\Windows\INetCacheContent.Word\NetworkUML.PNG

Figure 7: Network System UML Diagram

# TESTING AND IMPLEMENTATION

Testing will and always will be a difficult aspect of any development process. For this project, each subsystem will have their respective components tested through JUNIT testing before moving on to the next system. As of current, these tests have not been developed but it is clear what they should be; they will be created alongside (and before) actual system implementation and integration. Much of this will be updated as time persists.

## **UNIT TESTING**

Unit testing will be conducted for every object and system independently of each other. This will be done to ensure there is no issue with the algorithm or methodology used.

### **ENTITIES**

* Tiles
  + Have valid terrain
  + Rotations are valid
  + Can accurately check neighbors
* Farmland (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
  + Correctly update neighboring cities
* Roads (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
  + Correctly end road
* Monasteries (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
* Cities (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
  + Accurate shield status
  + Correct completion
* Board
  + Adds valid tiles
  + Correct storage of tiles
  + Correct retrieval of tiles
* Meeple
  + Correct placement
  + Correct player
* Players
  + Correct score
  + Correct meeple count
* Tile stack
  + Correct set of tiles
  + Acceptable randomization
  + Removal of invalid tiles
  + Correct alert when out

### **BOARD SYSTEM**

* Check that board was initialized
* Check that game was started
* Check that there are active players
* Check that there is a stack of tiles
* Check that regions are connected properly
* Check that tiles are placed
* Check that meeples are placed
* Check that all components are updated once tile is placed
* Check that scoring is correct
* Check that meeples are removed when regions are connected entirely

### **SCORING SYSTEM**

* forfeit case:
  + If player makes illegal move or runs out of time, system should declare the other player as winner.
* endgame cases:
  + if both players scores are same, both should be declared winner
  + if one player has greater score, they should be declared winner
* score during gameplay case:
  + if no meeples, it should not score terrain even if it is completed.
  + check if 'scorability' is defined right for each terrain type
  + check that farmer meeples don't get removed
  + check that calculated Points for terrain are correct

### **ACTION SYSTEM**

* Test if the system is able to set one player and one AI, two AIs, or two players
* Test if the system is able to set a player name
* Test if the system is able to place a random tile in the center of the board
* Test if the system is able to place a desired tile in a legal position of the board
* Test if the system is able to score the game properly
* Test if the system is able to move seamlessly to and from each action

### **ARTIFICIAL INTELLIGENCE SYSTEM**

* Check that best decision was made based on scenario
* Check that it is active

### **NETWORK SYSTEM**

* Check for valid connection (socket/port) ID
* Check for connected session
* Check for expected object format
* Check for data integrity
* Check that data was sent properly
* Check that data was received properly

## ACCEPTANCE TESTING

This will use a testing framework such as fitness to test use cases. This helps to ensure that integration is correctly done and that the different components interact correctly.

# APPENDICES

## APPENDIX A: CARCASSONNE GAME RULES

Appendix A is a printout of the Carcassonne’s game rules distributed by their publisher’s website at <http://riograndegames.com/getFile.php?id=670>