Requirements Engineering Artifact

I. Usage Model

a. Collection of Features

i. Game Logic

1. Enemies move side to side and slowly descend.
2. Enemies destroyed when struck by bullet.
3. Player destroyed when enemies reach player's ship.

ii. User Interaction

1. Left arrow moves player's ship left.
2. Right arrow moves player's ship right.
3. Spacebar fires bullet.

II. Initial Domain Model

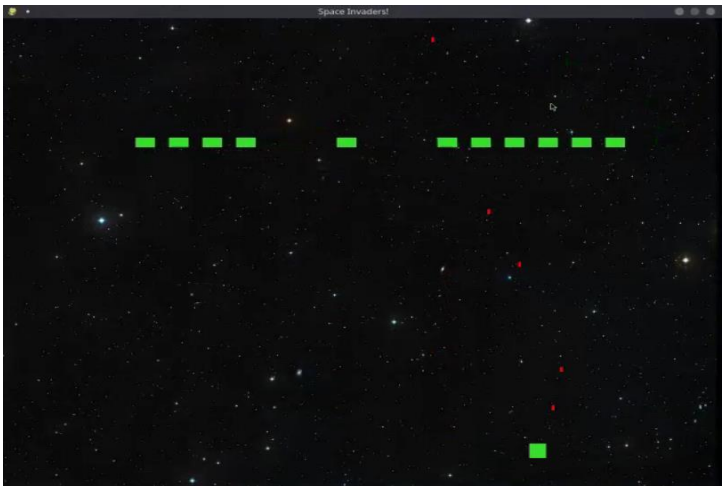
a. Class Responsibility Collaborator (CRC) Cards

Alien	
Position	Player
Dimensions	Bullet
Speed	
Move	

Player	
Position	Bullet
Dimensions	Alien
Speed	
Move	
Fire	

Bullet	
Position	Player
Dimensions	Alien
Speed	
Move	

III. User Interface Model



Estimation of Person-Hours

I. Project 1 Person-Hours
    a. Meetings
        i. (6 meetings) * (5 persons) * (0.5 hours) = 15 person-hours
    b. Individual Coding
        i. (1 person) * (5 hours) = 5 person-hours
        ii. (1 person) * (1 hours) = 1 person-hours
        iii. (1 person) * (4 hours) = 4 person-hours
        iv. (1 person) * (4 hours) = 4 person-hours
        v. (1 person) * (3 hours) = 3 person-hours
        vi. Total Person-Hours = 17 person-hours
    c. Group Coding
        i. (1 session) * (5 persons) * (4 hours) = 20 person-hours
    d. Total Person-Hours
        i. 15 + 17 + 20 = 52 person-hours

II. Project 2 Person-Hours
    a. Meetings
        i. (4 meetings) * (5 persons) * (0.5 hours) = 10 person-hours
    b. Individual Coding
        i. (1 person) * (3 hours) = 3 person-hours
        ii. (1 person) * (2 hours) = 2 person-hours
        iii. (1 person) * (2 hours) = 2 person-hours
        iv. (1 person) * (4 hours) = 4 person-hours
        v. (1 person) * (2 hours) = 2 person-hours
        vi. Total Person-Hours = 13 person-hours
    c. Group Coding
        i. (2 sessions) * (5 persons) * (5 hours) = 50 person-hours
    d. Total Person-Hours
        i. 10 + 13 + 50 = 73 person-hours

Estimation of Person-Hours (Continued)

Project 3 is more similar to Project 1 than Project 2. This is because both Project 3 and Project 1 involve coding a game from scratch in Python, whereas Project 2 involved enhancing a game in C++. Because of this, the estimated total person-hours for Project 3 will be closer to the total person-hours for Project 1. However, for Project 1, most of our group was not familiar with the Python language. For Project 3, most of our group now has a better understanding of Python and will be able to work more quickly. Taking this into consideration, a good estimate for the total person-hours required for Project 3 would be 50 person-hours.

Design Paradigm:

For this project, our team decided to follow the object-oriented design paradigm for our implementation of Space Invaders. One of the main reasons we chose to follow this design paradigm is because the language we used to develop our project, Python, is an object-oriented language, which made the decision easy. The object-oriented design was the best idea for our project because it was able to break down our other preferred design paradigms, Top-Down Functional Decomposition and Component-Level Design, into smaller, usable modules. The most useful part of the object-oriented design paradigm is the use of classes and objects to manage different parts of the program. Each component of the program was unique, and so being able to break them all up into tiny chunks so that they could be store information that was only relevant to its function was key. The use of classes also made it much more convenient in dividing up the work amongst our groups, as team members would be able to choose what component they work on. This allowed each person to really focus on one part of the code and made collaborating much easier, as you would speak with the person whose code you are going to connect with during this process. Another reason why we chose this design paradigm is because of how manipulatable the classes themselves were. We were easily able to give each class its bare functions so that we could get a prototype up and running, but this paradigm will allow us to easily add more features to each class as this project progresses.

Architecture:

Within the context of the design paradigm, the architecture of our prototype most closely resembles that of a 3-Tier architecture with a few exceptions. Because our program is a game, the architecture relies a lot on what might be considered the “Presentation Tier.” In this tier we have our user interface that translates what the user does to the game logic. Our display class that handles what is shown on screen easily falls within the “Presentation Tier.” Past this tier there exists our “logic” where every “event” that happens on the screen is handled. A lot of the logic is handled with game flow functions. These functions handle all sorts of things such as checking if there was a hit between the bullet and the enemy, checking if the game is over, and transitioning the player to the next level. The logic tier is also where a the user input is handled so that whenever there is a valid keystroke, the event is appropriately handles and displayed to the board. The logic tier interacts heavily between the presentation and data tiers. The data tier will is the lightest of all the tiers as it there is not much information being stored in the program. The information stored in the data tier includes information such as position on the screen for both

aliens and the player, as well as information regarding the display such as screen size and background images. The data class also hold information regarding the amount of aliens on the screen, so that the logic tier knows when the send the player to the next level.

Accounting of Actual Person-Hours

I. Turner Graham
   a. 10/22/20: 0.5 person-hours
   b. 10/24/20: 4 person-hours
   c. 10/25/20: 2 person-hours

II. Dawson Kabler
   a. 10/23/2020: 2 hours
   b. 10/24/20: 0.5 hours
   c. 10/25: 1.5 hours

III. Liam Connolly
   a. 10/21/20: 2
   b. 10/23/20:2
   c. 10/24/20: 3.5

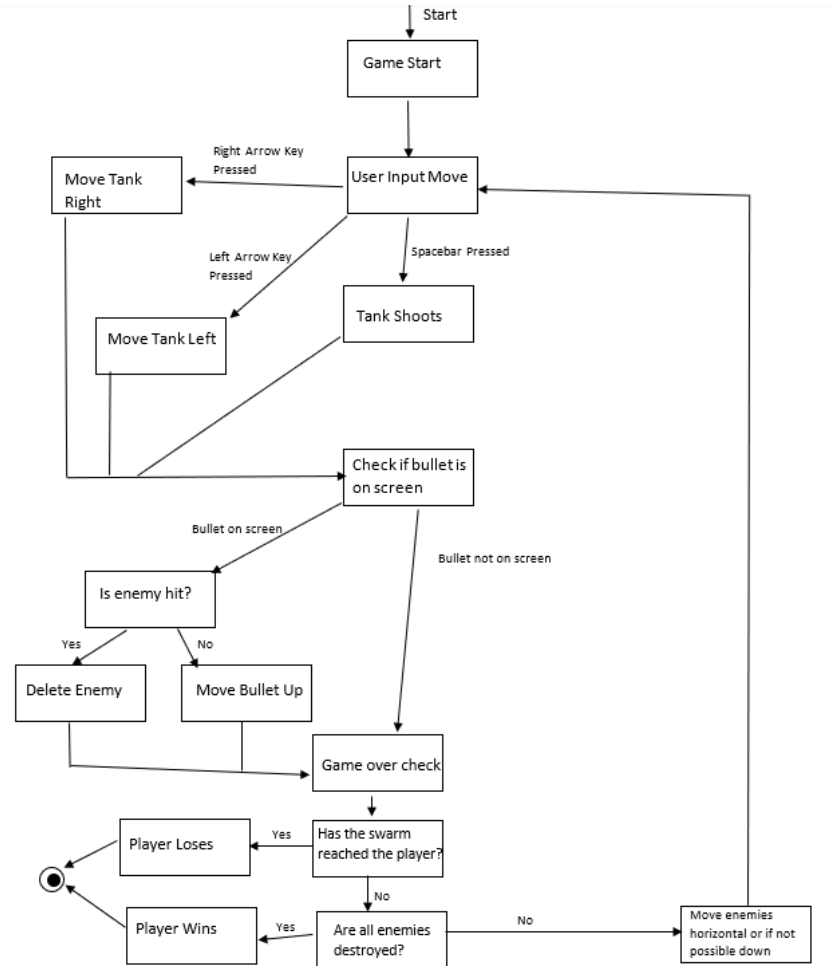
IV. Jarrett Zelif
   a. 10/19/20: 6
   b. 10/23/20: 1
   c. 10/24/20: 3

V. Carter Harrod
   a. 10/24/20: 6
   b. 10/25/20: 4
   c. 10/19/20: 0.5

VI. Total Person-Hours
   a.

VI. UML Project 3 Diagram

State Diagram for Project 3:



VII. Project 3 Design Pattern

The design pattern we used in the creation of our Project 3 prototype was iterator and flyweight. The reason iterator is one of our design patterns is that to create space invaders we needed multiple enemies in our game. We need to keep looping through this large group of enemies through frames to see if we had hit them with our bullet or if they were still unhit and could move horizontally on the next frame. It also was used in the same way for the bullet because we would continue to check with a specific delay time if we had hit an enemy with it, if there was no bullet currently in use, and if there was a bullet currently shot and has not hit anything yet. The reason flyweight was our other design pattern is because we had a large number of objects in our game such as enemies that were exactly the same which helped support our similar objects. This would be us being able to give the game flow class our entire army of aliens as to more efficiently handle the game instead of looking at a bunch if single enemies and this also helps us if we would want to make the swarm of enemies different sizes by being able to just fill the swarm with larger number of enemies. With these two design patterns were able to implement a more efficient and smoother running program that can be altered whenever our group would need to do so.