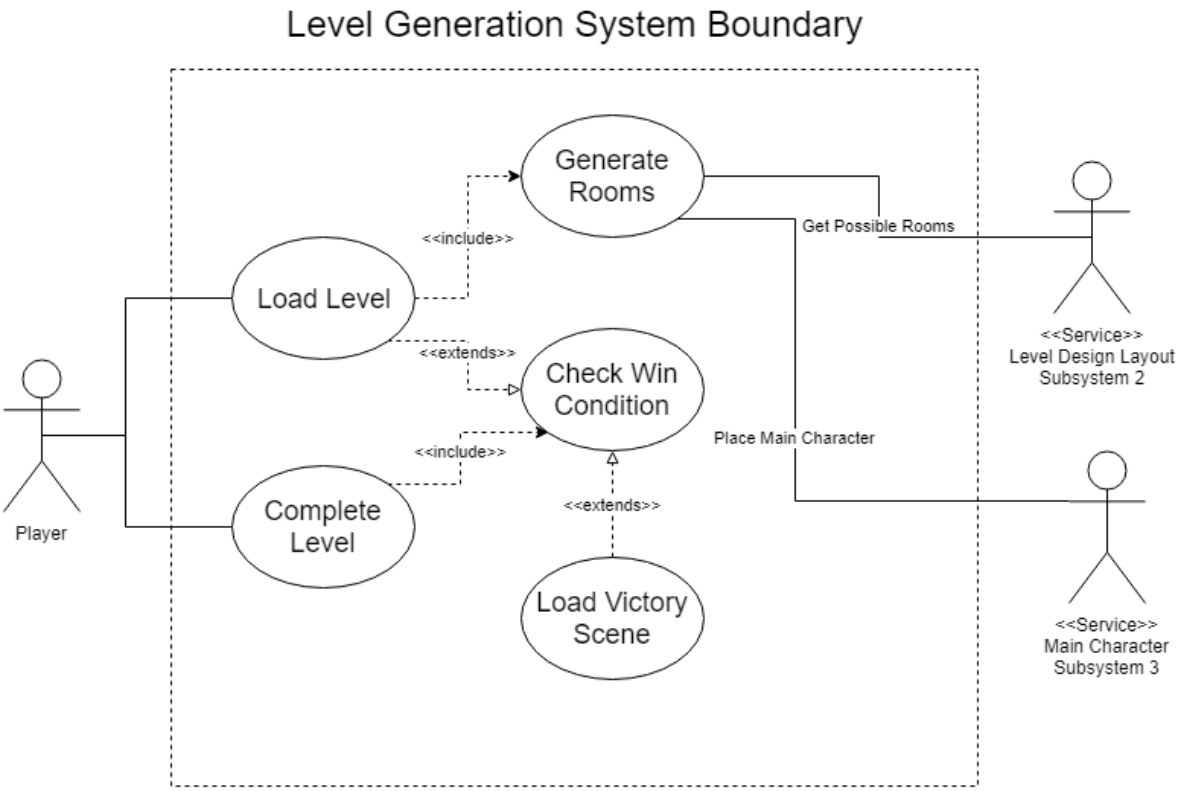
Zachary Sugano
Dr. Beeston
CS 383
07 February 2019

Champion

1. **Brief Introduction ___/3**
   The feature that I am going to create is procedural level generation. We want to give the player a unique experience every time that they play through the game and so the levels of the dungeon will be generated randomly. We are going to create a variety of prefabricated rooms that we will then stitch together to form an entire level.

2. **Use case diagram with scenario ___/14**



Scenarios:

1.1 Load Level

Summary: The player loads a new level either by starting the game or completing a previous level

Actors: Player

Preconditions: Player has loaded the game and started playing. Difficulty variable has been initialized.

Basic Sequence:

1. Player loads level by starting game or completing a previous level
2. A list of potential rooms are received from the Level Design Layout subsystem.
3. Rooms are generated using the difficulty variable, the list of rooms, and a random number generator

Exceptions: None, if the load level use case is triggered then rooms WILL be generated. There is no other potential extraneous behavior that could occur conditionally that would modify this behavior.

Post Conditions: Rooms are generated. Player character is placed in the start room and the level is rendered to the player.

Priority: 1

ID: ZS01

1.2 Complete Level

Summary: The player completes a level by reaching the end of the series of rooms

Actors: Player

Preconditions: Player has loaded the game and started playing. Level has already been generated and the player has reached the end

Basic Sequence:

1. Player reaches the end of a level
2. Check the win condition

Exceptions:

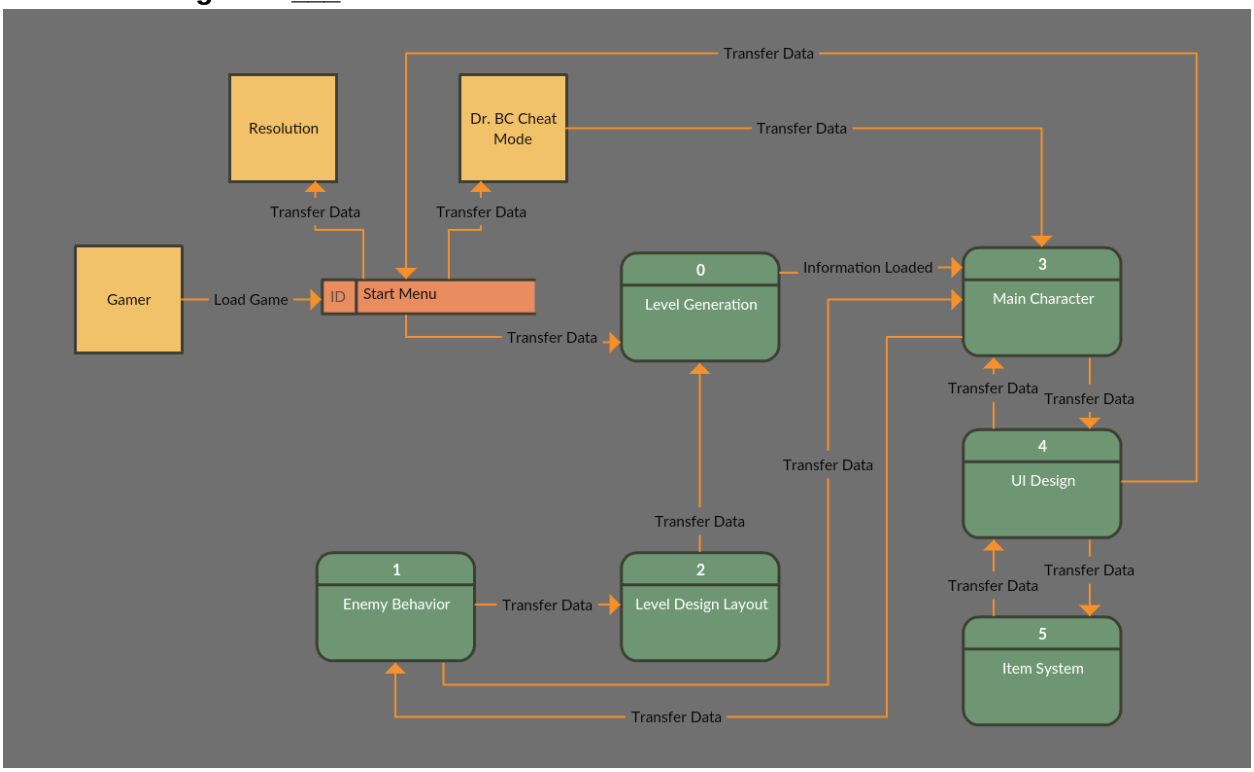
1. Player has satisfied all win conditions for the game, load victory scene.
2. Player has not satisfied all win conditions for the game, they advance to the next level and the load level use case is triggered.

Post Conditions: The player either enters the next level or is sent to the victory screen.

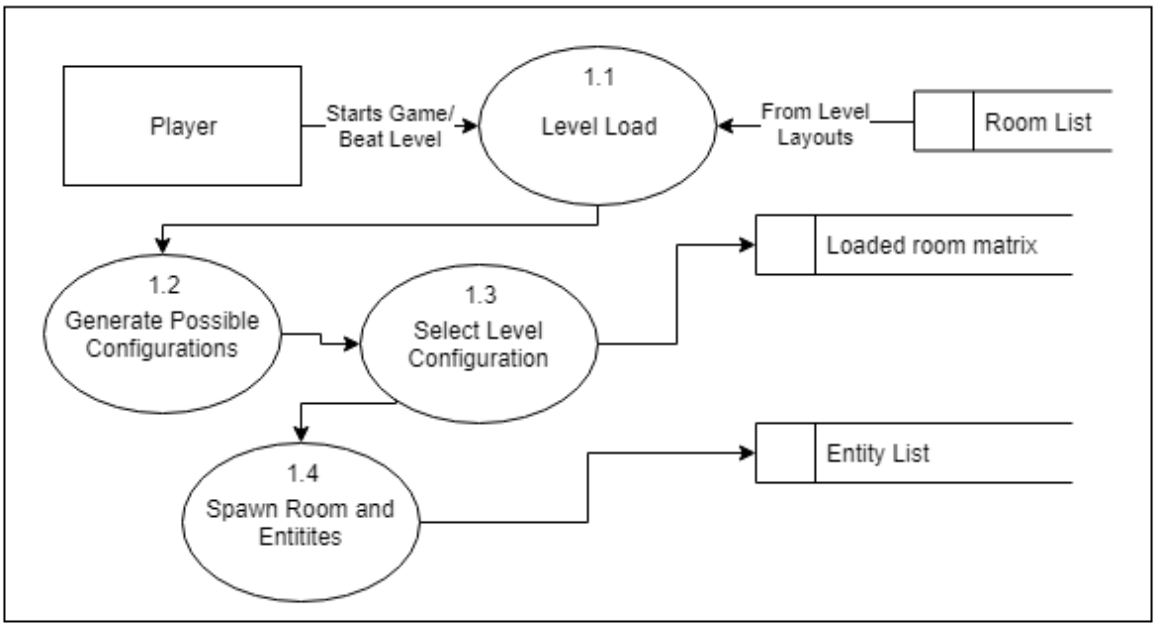
Priority: 1

ID: ZS02

3. Data Flow Diagrams ___/14



Process 0 Level Generation Diagram



Process Description:

1.1 Level Load:

- Load Level function is called by player starting game or beating previous level
- Get the list of all possible rooms from the Level Design Layout Module

1.2 Generate Possible Configurations

- Using list of possible rooms and a random number generator, generate a matrix of possible room layouts.
- Pass the list of possible room matrices to the test function in 1.3

1.3 Select Level Configuration

- Take list of possible room matrices and use a function to score each room matrix
- Take the highest scoring rooms and select a random one to store in the loaded room matrix. This info is passed to 1.4

1.4 Spawn Room and Entities

- Using the room matrix, spawn each room and the associated entities
- Store spawned entities in entity list

4. **Acceptance Tests ___/9**

Random level winnability test:
1. Run level generation algorithm
2. Check if there is a navigable path from start to finish
3. Repeat a large number of times to verify that each level is beatable with 100% success rate.

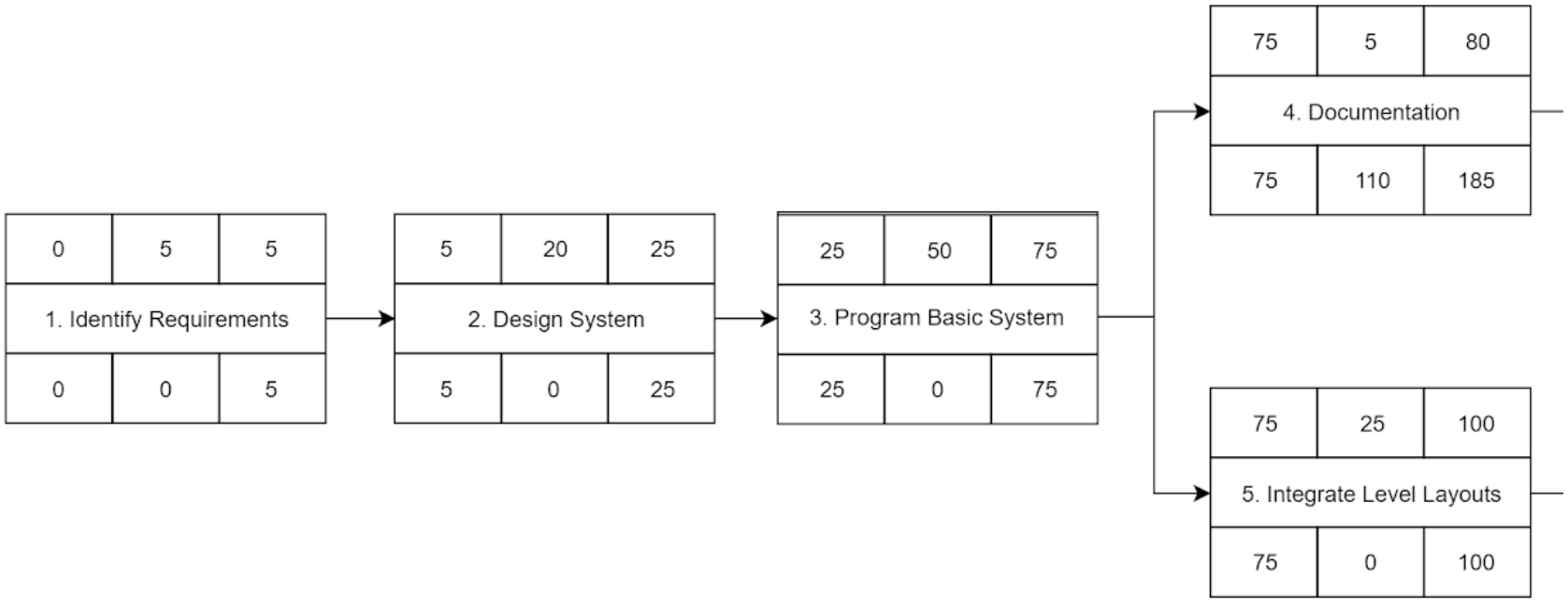
Level variability test:
1. Run level generation algorithm
2. Iterate over all rooms in the level and check that no two rooms of the same type are adjacent to one another.
3. Repeat a large number of times to verify that each level has enough variety with 100% success rate.

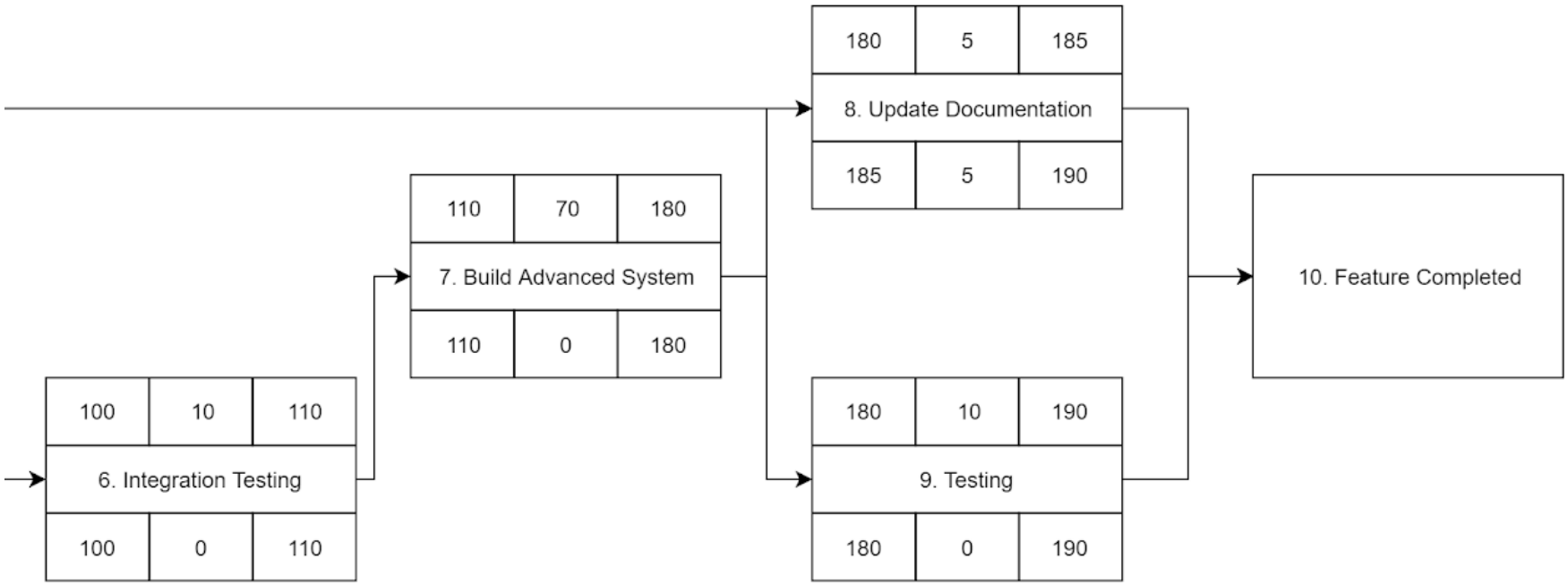
Entity generation test:
1. Run level generation algorithm
2. Iterate over all rooms in a level and check that each room has spawned the correct entities such as enemies, items, and quest markers needed to complete the level.
3. Check that the win condition for the level is doable if all entities are killed and all items are collected.

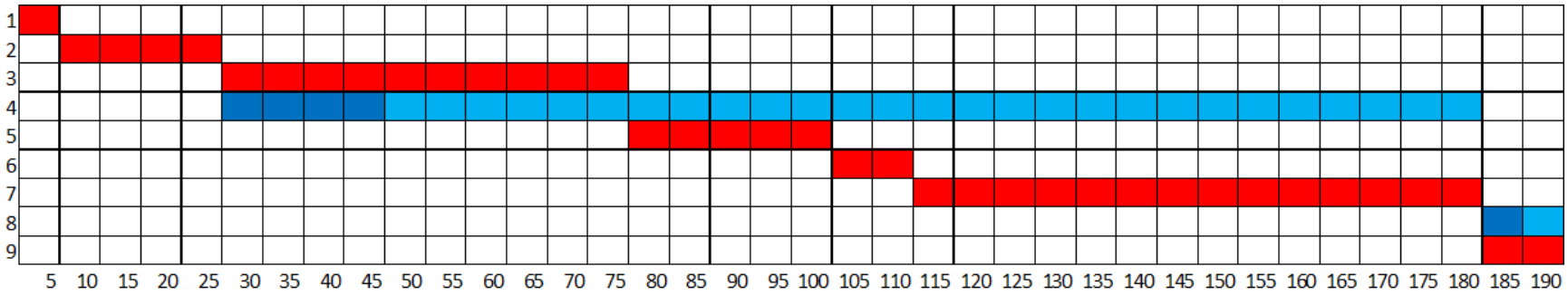
5. **Timeline __/10**

Task	**Duration (hrs)**	**Predecessors**

1. Identify requirements	5	N/a
2. Design system	20	1
3. Program basic system	50	2
4. Documentation	5	3
5. Integrate Level Layouts	25	3
6. Integration Testing	10	5
7. Build advanced system	70	6
8. Update Documentation	5	7, 4
9. Testing	10	7


0
5
5
1. Identify Requirements
0
0
5
5
20
25
2. Design System
5
0
25
25
50
75
3. Program Basic System
25
0
75
75
5
80
4. Documentation
75
110
185
75
25
100
5. Integrate Level Layouts
75
0
100


180
5
185
8. Update Documentation
185
5
190
110
70
180
7. Build Advanced System
110
0
180
10. Feature Completed
100
10
110
6. Integration Testing
100
0
110
180
10
190
9. Testing
180
0
190


1
2
3
4
5
6
7
8
9
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190