Max Williams

1/29/2017

Assignment 1

1. Problem 1.1

The basic tasks that all software engineering must handle are requirement gathering, high-level design, low-level design, development, testing, deployment, maintenance, and wrap-up.

2. Problem 1.2

Requirement gathering is the process of determining what the customer's needs are, and turning them into requirement documents which tell the customer what to expect in the finished product and keep developers on track. High-level design is focused on determining the main components and how they will interact with each other. Low-level design then breaks down and describes how each of the main components will work and what components they will contain. Development is the actual programming based on the plans laid out in the previous tasks. Testing is where bugs and issues within the existing product are identified and fixed in a series of iterations because fixing one bug may create more. Deployment is setting up the product so that end-users can begin using it, this may require setting up hardware and training new users. Maintenance is the task of responding to any issues or bugs users identify in the product and ensuring its quality. Finally, wrap-up is postmortem evaluation of how the project went well and how it failed in order to improve for the next project.

3. Problem 2.4: Completed

4. Problem 2.5

JBGE stands for "Just Barely Good Enough" which is primarily meant to apply to code comments and documentation. The concept is that while documentation is important, why waste time writing excessive documentation when you could be working on something else. However, this approach can get out of hand quickly, creating unmaintainable code that no-one understands. Additionally, this approach can also be applied to requirement documents, leading to a product that strays from customer expectations.

5. Problem 3.2
    a. 5 days till completion
    b. 9 days till completion
    c. 4 days till completion
    d. 12 days till completion
    e. 19 days till completion
    f. 7 days till completion
    g. 6 days till completion
    h. 3 days till completion
    i. 6 days till completion
    j. 6 days till completion
    k. 17 days till completion
    l. 12 days till completion
    m. 28 days till completion
    n. 26 days till completion
    o. 11 days till completion

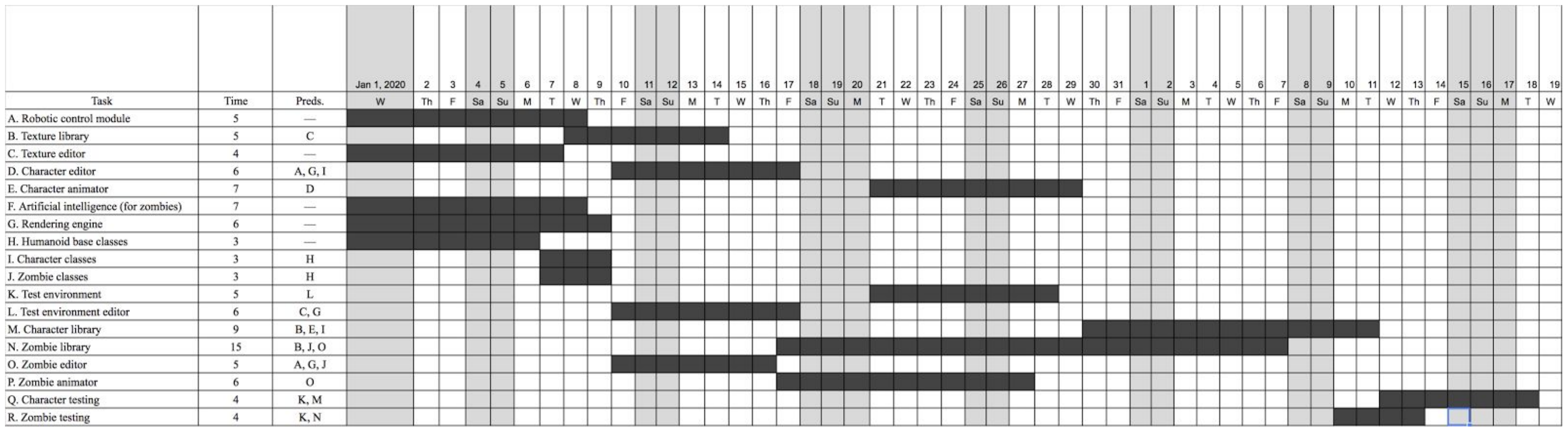
p. 17 days till completion

q. 32 days till completion

r. 30 days till completion

The critical paths for this project is G → D → E → M → Q and H → I → D → E → M → Q, both of which take 32 days total. Therefore, the total working days on this project can be expected to be 32 days.

6. Problem 3.4



			Jan 1, 2020	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Task	Time	Preds.	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W
A. Robotic control module	5	—	■	■	■	■	■	■	■	■																																										
B. Texture library	5	C								■	■	■	■	■	■	■																																				
C. Texture editor	4	—	■	■	■	■	■	■	■																																											
D. Character editor	6	A, G, I										■	■	■	■	■	■	■	■																																	
E. Character animator	7	D																					■	■	■	■	■	■	■	■	■																					
F. Artificial intelligence (for zombies)	7	—	■	■	■	■	■	■	■	■																																										
G. Rendering engine	6	—	■	■	■	■	■	■	■	■	■																																									
H. Humanoid base classes	3	—	■	■	■	■	■	■																																												
I. Character classes	3	H							■	■	■																																									
J. Zombie classes	3	H							■	■	■																																									
K. Test environment	5	L																					■	■	■	■	■	■	■	■																						
L. Test environment editor	6	C, G										■	■	■	■	■	■	■	■																																	
M. Character library	9	B, E, I																														■	■	■	■	■	■	■	■	■	■	■	■	■								
N. Zombie library	15	B, J, O																	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■												
O. Zombie editor	5	A, G, J										■	■	■	■	■	■	■																																		
P. Zombie animator	6	O																	■	■	■	■	■	■	■	■	■	■	■																							
Q. Character testing	4	K, M																																										■	■	■	■	■	■	■	
R. Zombie testing	4	K, N																																								■	■	■	■						

7. Problem 3.6

In order to account for unexpected problems teams can use strategies such as adding an additional 5% to the time for a task, or add a task to the project's Gantt chart and use it whenever an unexpected delay occurs.

8. Problem 3.8

One of the biggest mistakes you can make while tracking tasks are to overemphasize the importance of deadlines, causing developers to claim they are on track with their task while they are actually behind schedule. The other is to assign additional developers to a task when it is behind schedule, while an extremely knowledgeable developer may help with the task it is more likely that getting them up to speed on the task will require more time than sticking with the initial developers.

9. Problem 4.1

Good requirements are clear, unambiguous, consistent, prioritized, and verifiable.

10. Problem 4.3

a. Allow users to monitor uploads/downloads while away from the office.

- Business

b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password.

- User, Functional

c. Let the user specify upload/download parameters such a number of retries if there's a problem.

- User, functional

d. Let the user select an Internet location, a local file, and a time to perform the upload/download.

- User, functional

e. Let the user schedule uploads/downloads at any time.

- User, functional

f. Allow uploads/downloads to run at any time.

- Functional

g. Make uploads/downloads transfer at least 8 Mbps.

- Non-functional

h. Run uploads/downloads sequentially. Two cannot run at the same time.

- Functional

i. If an upload/download is scheduled for a time when another is in progress, it waits until the other one finishes.

- Functional

j. Perform schedule uploads/downloads.

- Business

k. Keep a log of all attempted uploads/downloads and whether the succeeded.

- Functional

l. Let the user empty the log.

- User, functional

m. Display reports of upload/download attempts.

- User, functional

n. Let the user view the log reports on a remote device such as a phone.

- User, functional

o. Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times.

- User, functional

p. Send a text message to an administrator if an upload/download fails more than it's maximum retury number of times.

- User, functional

This project doesn't have any implementation requirements, or temporary requirements that are needed to transition into using the new system. While this project does require error notifications those requirements are permanent components of the application and therefore not implementation requirements.

11. Problem 4.9

MOSCOW: Must, Should, Could, Wont

<u>Must:</u>

- Save current game progress when the application is closed.
- Prompt users with a confirmation window before starting a new game.

<u>Should:</u>

- Allow for a second user to enter a word for another to guess, effectively making the game two player.
- Compare user-entered word with the existing dictionary to determine if valid.
- If a word has been played prevent it from being used in the next 10 games.

<u>Could:</u>

- Have a remaining bones counter which tells the user how many incorrect guesses they have left.
- Add user accounts.
- Track the performance of each player on their account.
- Implement a leaderboard listing the most games won, longest win streak, and most games played.

<u>Wont:</u>

- Have a graphical representation of the bones remaining to identify how many incorrect guesses are left.

- Make a new bone flash/glow when added to Mr. Bone.