Nicholas Soffa

Professor Johnson

CMSI 402

30 January 2017

Assignment #1

1.1 What are basic tasks that all software engineering projects must handle?

Discovering Requirements, High-Level Design, Low-Level Design, Development, Testing, Deployment, and a Project Wrap-Up.

1.2 Give a one sentence description of each of the tasks you listed in Question 1.

Discovering Requirements is the beginning stage of deciding what will be needed in order to create the project. High-Level Design is the stage of choosing the platform or data design to use for the project. Low-Level Design is deciding how the project is going to work when put into use, like deciding how many columns will be in a data table in your project. Development is the stage of developing the project itself. Testing is the stage of going through the project and testing things out while looking for bugs. Deployment is actually getting the project out to the expected user base. Lastly, the project wrap-up is to look at what went wrong and what went right during the project’s development.

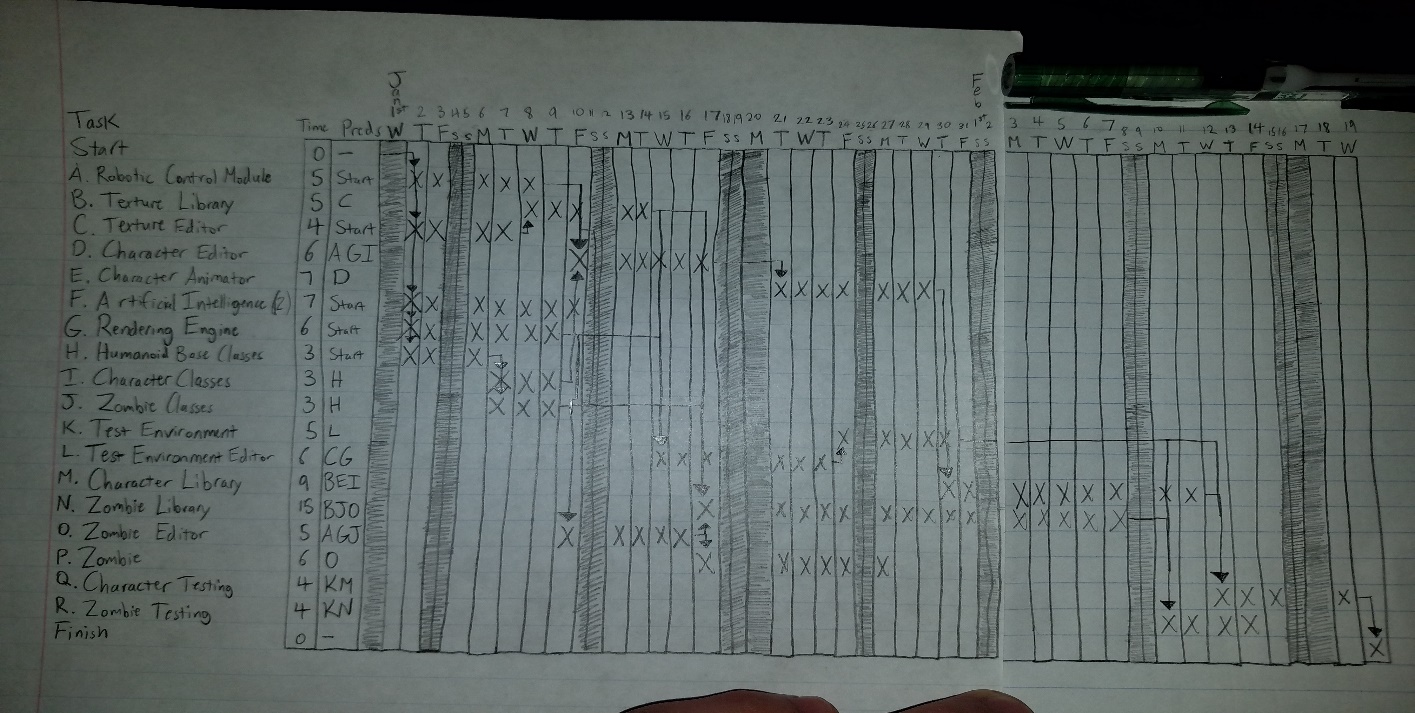
2.5 What does JBGE stand for and what does it mean?

Just barely good enough. The idea behind it is that you should only provide enough code documentation and comments that are “just barely good enough” because during updates and fixes to code you will spend too much time updating the comments as well if you put too much effort into them.

3.2 Use critical path methods to find the total expected time from the project’s start for each task’s completion. Find the critical path. What are the tasks on the critical path? What is the total expected duration of the project in working days?

C:\Users\NICK\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Untitled Diagram.png

The Critical Path is Start->G->D->E->M->Q->Finish. The total expected duration is 32 days.

3.4 Build a Gantt chart for the network you drew in 3.2. 

3.6 In addition to losing time from vacation and sick leave, projects can suffer from problems that just strike out of nowhere. Sort of a bad version of *deus ex machina*. For example, senior management could decide to switch your target platform from Windows desktop PSs to the latest smartwatch technology. Or a strike in the Far East could delay the shipment of your new servers. Or one of your developers might move to Iceland. How can you handle these sorts of completely unpredictable problems?

I would account for vacation times and delay times and add those estimates in to the end project date in order to be as close to the final date as possible.

3.8 What are the two biggest mistakes you can make while tracking tasks?

The two biggest mistakes you can make are ignoring a problem and hoping you can make up time later, and throwing more people at a problem and assuming that will hasten the process.

4.1 List five characteristics of good requirements

Clear, Unambiguous, Consistent, Prioritized, and Verifiable.

4.3 Suppose you want to build a program called TimeShifter to upload and download files at scheduled times while you’re on vacation. The following list shows some of the applications requirements. For this exercise, list the audience-oriented categories for each requirement. Are these requirements in each category?

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

User Requirement

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

User Requirement

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

User Requirement

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

User Requirement

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

User Requirement

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

Nonfunctional Requirement

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

Nonfunctional Requirement

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

Nonfunctional Requirement

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

Functional Requirement

j. Perform schedule uploads/downloads.

User Requirement

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

Functional Requirement

l. Let the user empty the log.

User Requirement

m. Display reports of upload/download attempts.

User Requirement

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

User Requirement

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

Functional Requirement

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

Functional Requirement

4.9 Figure 4-1 [right] shows the design for a simple hangman game that will run on smartphones. When you click the New Game button, the program picks a random mystery word from a large list and starts a new game. Then if you click a letter, either the letter is filled in where it appears in the mystery word, or a new piece of Mr. Bones's skeleton appears. In either case, the letter you clicked is grayed out so that you don't pick it again. If you guess all the letters in the mystery word, the game displays a message that says, "Congratulations, you won!" If you build Mr. Bones's complete skeleton, a message says, "Sorry, you lost." Brainstorm this application and see if you can think of ways you might change it. Use the MOSCOW method to prioritize your changes.

I would definitely include a retry button after you have won or lost the game, and then could/won’t include a dancing animation for the skeleton if you lose.