Simon Wroblewski

Dr. Johnson

402 Thesis Project

Reading Assignment #1

1.1

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

* Gathering requirements
* High- and Low-level design
* Development
* Testing
* Deployment
* Maintenance
* Wrap-up

1.2

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

* Requirements gathering refers to collecting data about the target demographics wants and needs.
* High-Level design refers a broad overview of how the software and how it interacts.
* Low-Level Design provides a more detailed explanation of the software and its inner workings.
* Development the writing/creating/producing of the software pieces that will come together to create an application that meets all the specified criteria.
* Testing is a process where the programmer puts the software under multiple conditions, some strenuous, to ensure the application is robust and has as few possible malfunctions as possible.
* Deployment is the process where the application is made available to the target demographic.
* Maintenance the application is reviewed given past data collection and updates are installed to either increase functionality or to address certain bugs or malfunctions.
* Wrap-up refers to the time when the entire project is analyzed from start to finish so that that we can see where we fell short and where we excelled.

2.4

Like Microsoft Word, Google Docs [sic] provides some simple change tracking tools. Go to <http://www.google.com/docs/about/> to learn more and sign up [if you do not have an account already]. Then create a document, save it, close it, reopen it, and make changes to it as you did in Exercise 1. **Complete**

2.5

What does JBGE stand for and what does it mean?

* JBGE is an acronym for Just Barely Good Enough and it its used commonly as a comment in code referring to the quality of the code being sub-par and only slightly reaching requirements

3.2 and 3.4

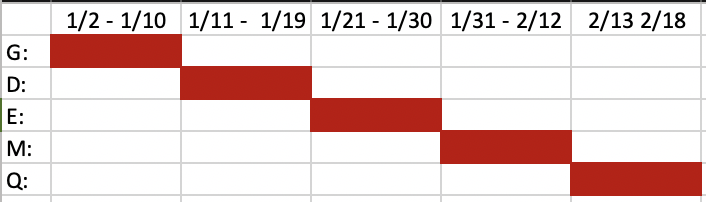
Use critical path methods to find the total expected time from the project's start for each task's completion. Find the critical path. What are the tasks on the critical path? What is the total expected duration of the project in working days?

* Critical Path:

> G: Rendering Engine (8 days) -> D: Character editor (8 days) -> E Character animator (8 days) -> M: Character Library (11 days) -> Q: Character Testing (5 days)

> 8 + 8 + 9 +11+ 5 = 31 days

* Gant Chart:



This Gant chart represents the critical path; vacations, holidays, and any other lost time can be shown here visually.

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?

* This problems can be handled in many ways, such as providing extra training to reduce these unforeseen problems or even breaking up large tasks into smaller, more manageable ones.

3.8

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

* Throwing more people at the problem under the guise “Many hands make lighter work”
* Omitting or forgetting to update the current project so everyone is on the same page

4.1

List five characteristics of good requirements.

* Unambiguous, Clear, Prioritized, Consistent, Verified

4.3

TimeShifter Program:

* Allow users to monitor uploads/downloads while away from the office. [Business]
* Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password. [Business]
* Let the user specify upload/download parameters such a number of retries if there's a problem. [Functional, User]
* Let the user select an Internet location, a local file, and a time to perform the upload/download. [Functional, User]
* Let the user schedule uploads/downloads at any time. [Not Functional]
* Allow uploads/downloads to run at any time. [Not Functional]
* Make uploads/downloads transfer at least 8 Mbps. [Not Functional]
* Run uploads/downloads sequentially. Two cannot run at the same time. [Not Functional]
* If an upload/download is scheduled for a time while another is in progress, it waits until the other one finishes. [Not Functional]
* Perform schedule uploads/downloads. [Functional]
* Keep a log of all attempted uploads/downloads and whether the succeeded. [Functional]
* Let the user empty the log. [Functional, User]
* Display reports of upload/download attempts. [Functional, User]
* Let the user view the log reports on a remote device such as a phone. [Functional, User]
* Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times. [Functional, User]
* Send a text message to an administrator if an upload/download fails more than it's maximum return number of times. [Functional, User]

4.9

Moscow changes: (MuSCoW or Must, Should, Could, Won’t)

* Must: find a way to be reimbursed for your time
* Should: Create a platform to make an account and play with friends, allowing one friend to choose the word and another to guess
* Could: Add more than just hangman to the application and make it a platform for multiple table top games
* Won’t: Change the functionality so much that it becomes an entirely new application all together, thus abandoning its original idea