Arash Lari

Homework 1

CMSI 402

Prof. B.J. Johnson

Problem 1.1, Stephens page 12

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

Answer: Requirements Gathering, High-level Design, Low-level Design, Development, Testing, Deployment, Maintenance, Wrap-Up

Problem 1.2, Stephens page 12

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

Answer:

Requirements Gathering: Figure out what the customer wants

High Level Design: At a high level, describe the major pieces of the application and how they interact

Low-Level Design: Get into the details about how to specifically build the pieces of the application so the programmers can implement it

Development: The physical act of writing code to implement the application

Testing: Stress test the application under different circumstances to try to detect and prevent flaws or bugs before deployment

Maintenance: Bug fixes, additions, enhancements, adding future versions of the program

Wrap-up: Look at everything that’s been done in the projects development history to figure out what went right and what went wrong to learn from each project to better development of future projects

Problem 2.4, Stephens page 26

**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.**

Answer: I did it, I’m not sure what to do to prove it. I wrote this on a google doc.

Problem 2.5, Stephens page 26

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

Answer: Just barely good enough, it basically means that you shouldn’t write any more code documentation or comments than absolutely necessary.

Problem 3.2, Stephens page 51

**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?**

Answer: The critical path passes through the tasks G, D, E, M, and Q. The total expected length of working days is 32. The tasks on the critical path are: Rendering Engine, Character editor, Character animator, Character Library, Character Testing.

Problem 3.4, Stephens page 51

**Build a Gantt chart for the network you drew in Exercise 3. [Yes, I know, you weren't assigned that one — however, when you do Exercise 2 you should have enough information for this one.] Start on Wednesday, January 1, 2020, and don't work on weekends or the following holidays:**

**Holiday Date**

**New Year's Day January 1**

**Martin Luther King Day January 20**

**President's Day February 17**

Answer: The game should finish by the end of day February 18th.

Problem 3.6, Stephens page 51

**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?**

Answer: You could treat sick leaves and deus ex machine problems the same way; by adding tasks at the end of the schedule to account for completely unexpected problems occur and when those problems do occur, insert its lost time into the schedule.

Problem 3.8, Stephens page 51

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

Answer: The biggest mistake you can make while tracking tasks is not taking action when a task slips. You must pay close attention to tasks that slip to know if it’s in trouble and whether or not you can take action. The second biggest mistake is adding more people on a task and assuming that it will cut down the total time. Unless the new additions to the team have a particularly special set of skills, it will usually just add more time to the time required to complete the task.

Problem 4.1, Stephens page 82

**List five characteristics of good requirements.**

Answer: Easy to understand, unambiguous, consistent, prioritized,verifiable

Problem 4.3, Stephens page 82

**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.**

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

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

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

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

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

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

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

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

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

**j. Perform schedule uploads/downloads.**

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

**l. Let the user empty the log.**

**m. Display reports of upoad/download attempts.**

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

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

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

**For this exercise, list the audience-oriented categories for each requirement. Are there requirements in each category? [If not, state why not…]**

Answer:

A: Business. B: User, Functional. C. User, Functional. D. User, Functional. E. Nonfunctional F. Nonfunctional G. Nonfunctional H. Nonfunctional I. Nonfunctional J. Functional. K. Functional. L. User, Functional. M. User, Functional. N. User, Functional. O. User, Functional. P. User, Functional.

All the categories have at least one requirement except for implementation requirements, which is empty. They might need new hardware to support the application, but if they’re performing uploads and downloads already then they probably don’t need to implement anything.

Problem 4.9, Stephens page 83-84

**Figure 4-1 [right] shows the design for a simple hangman game that will run on smartphones. When you click the New Bame 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, "Contratulations, 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.**

Answer: MOSCOW method stands for M: Must (needed). S: Should (Important). C: Could (desirable) W: Won’t (completely optional).

The musts include: Advertising because for an application to survive they either normally charge a fee to download or they display ads, which this doesn’t do currently. Monetization is important for applications’ survival.

The Shoulds include: Scoring and Score Keeping, high scores; having a simple win and lose game could go stale, having scores and keeping track could incentivize the user to use the application more than they would’ve to try to beat their previous scores.

The Coulds: Different fonts, themes, etc. Would be nice and add some variety to the screen. Skill levels would add variety and challenge players.

The Wonts: Online High scores, to compete with friends. Different languages, for users from different countries. Different ways of interacting rather than just typing (drag and drop etc.).