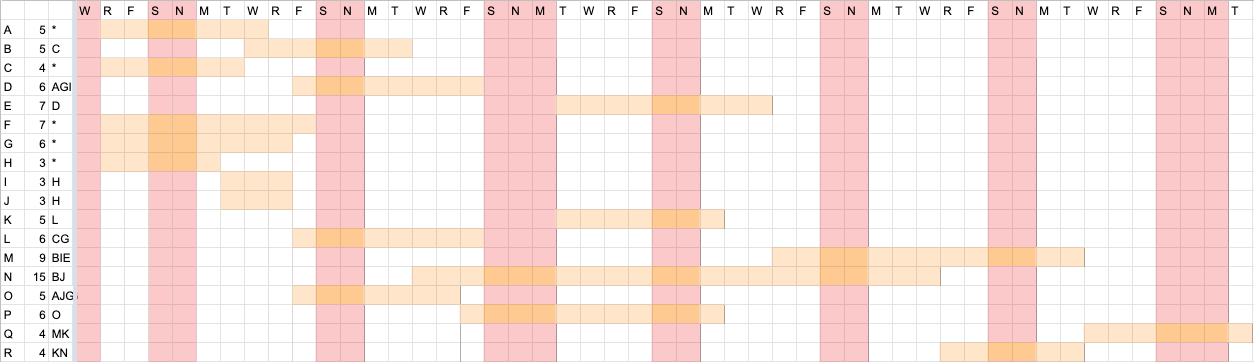
Ben Kern

CMSI 402

Dr. Johnson

HW #1

1. What are the basic tasks that all software engineering projects must handle?
   1. Requirements Gathering
   2. High-Level Design
   3. Low-Level Design
   4. Development
   5. Testing
   6. Deployment
   7. Maintenance
   8. Wrap-up
2. Give a one sentence description of each of the tasks you listed in Exercise 1.
   1. Requirements Gathering—Learn the customer’s wants and needs.
   2. High-Level Design—Describe the major pieces of the application and how they interact.
   3. Low-Level Design—Provide more detail about how to build the pieces of the application so that the programmers can actually implement them.
   4. Development—Write code to implement the application.
   5. Testing—Use the application under different circumstances to try to detect any flaws or bugs.
   6. Deployment—Roll out the application to the users.
   7. Maintenance—Implement bug fixes, additions, enhancements, and future versions of the program.
   8. Wrap-up—Evaluate the project’s history to determine what went right and what went wrong so that you can repeat the good things and avoid the bad things in future projects.
3. 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.
   1. Check mark
4. What does JBGE stand for and what does it mean?
   1. JBGE stands for Just Barely Good Enough. It’s the philosophy that you shouldn’t write any more code documentation or comments than absolutely necessary.
5. 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?
   1. The critical path passes through the tasks G, D, E, M, and Q and has a total expected length of 32 working days.
6. 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:
7. 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?
   1. You can treat deus ex machina problems the same way you handle unexpected sick leave. Add tasks at the end of the schedule to account for completely unexpected problems. When one of these problems does occur, insert its lost time into the schedule.
8. What are the two biggest mistakes you can make while tracking tasks?
   1. The biggest mistake you can make while tracking tasks is not taking action when a task slips. At a minimum, you need to pay closer attention to the task so that you can take action if it’s in trouble.
   2. The second biggest mistake is piling more people on the task and assuming they can cut the total time. Unless the new people have particularly useful expertise, bringing them up to speed may make the task take even longer.
9. List five characteristics of good requirements.
   1. Five characteristics of good requirements are clear, unambiguous, consistent, prioritized, and verifiable.
10. Suppose you want to build a program called TimeShifter to upload and download files at scheduled times while you're on vacation. For this exercise, list the audience-oriented categories for each requirement. Are there requirements in each category? [If not, state why not…]
    1. Allow users to monitor uploads/downloads while away from the office. (B)
    2. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password. (U, F)
    3. Let the user specify upload/download parameters such as number of retries if there’s a problem. (U, F)
    4. Let the user select an Internet location, a local file, and a time to perform the upload/download. (U, F)
    5. Let the user schedule uploads/downloads at any time. (N)
    6. Allow uploads/downloads to run at any time. (N)
    7. Make uploads/downloads transfer at least 8 Mbps. (N)
    8. Run uploads/downloads sequentially. Two cannot run at the same time. (N)
    9. If an upload/download is scheduled for a time when another is in progress, the new task waits until the other one finishes. (N)
    10. Perform scheduled uploads/downloads. (F)
    11. Keep a log of all attempted uploads/downloads and whether they succeeded. (F)
    12. Let the user empty the log. (U, F)
    13. Display reports of upload/download attempts. (U, F)
    14. Let the user view the log reports on a remote device such as a phone. (U, F)
    15. Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times. (U, F)
    16. Send a text message to an administrator if an upload/download fails more than its maximum retry number of times. (U, F)
11. 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.
    1. Animations (C)—When the user wins or loses, or if the program is in idle, the program could display an animation.
    2. Customizable GUI (C)—Allow the users to change the font or color of the GUI to their liking.
    3. Quick win (C)—The program could allow the user to type a guess for the whole word to get extra points.
    4. Multiple skill levels (C)—The program could allow users to pick a skill level. An algorithm would use word length and the letters in a word to estimate difficulty.
    5. Time limits (W)—The program could display a countdown. Each correct guess would increase the time available.