Ronald Uy
CMSI 402 - Assignment 1

Chapter 1

1. What are the basic tasks that all software engineering projects must handle?
   Requirements gathering
   High-level design
   Low level design
   Development
   Testing
   Deployment
   Maintenance
   Wrap-Up
2. Give a one sentence description of each.
   a. *Requirements Gathering*: Requirements gathering is to learn the customer's wants and needs for a program.
   b. High Level Design: High level design is to describe the major pieces of the program/application and how they interact.
   c. Low Level Design: Low level design is similar to an instruction set on how to build the pieces of the application so the programmer can implement them.
   d. Development: Development is actually writing the code to implement the application.
   e. Testing: Testing is to use the application under different circumstances to find any flaws or bugs.
   f. Deployment: Deployment is rolling out the application to the users.
   g. Maintenance: Maintenance is implementing any bug fixes, additions, enhancements, and future versions of the program.
   h. Wrap-Up: Wrap-up is evaluating the project's history to determine the good and the bad aspects so that future projects will avoid any of the bad aspects.

Chapter 2:

1.
2. What does JBGE stand for and what does it mean?
   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.

Chapter 3:

1. Finding the total expected time using critical path methods based on the figure, the critical path follows G > D > E > M > Q with a total expected time being about 32 days.
2. <will include graph>

3. 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.
4. 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. 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.

Chapter 4:

1. Five characteristics of good requirements are clear (easy to understand), unambiguous, consistent, prioritized, and verifiable.
2. The following lists show the TimeShifter requirements with their audience-oriented categories shown in parentheses. (B = Business, U = User, F = Functional, N = Nonfunctional, and I = Implementation.)

    a. Allow users to monitor uploads/downloads while away from the office. (B)
    b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password. (U, F)
    c. Let the user specify upload/download parameters such as number of retries if there's a problem. (U, F)
    d. Let the user select an Internet location, a local file, and a time to perform the upload/ download. (U, F)
    e. Let the user schedule uploads/downloads at any time. (N)
    f. Allow uploads/downloads to run at any time. (N)
    g. Make uploads/downloads transfer at least 8 Mbps. (N)
    h. Run uploads/downloads sequentially. Two cannot run at the same time. (N)
    i. If an upload/download is scheduled for a time when another is in progress, the new task waits until the other one finishes. (N) Chapter 4 | 369
    j. Perform scheduled uploads/downloads. (F)
    k. Keep a log of all attempted uploads/downloads and whether they succeeded. (F)
    l. Let the user empty the log. (U, F)
    m. Display reports of upload/download attempts. (U, F)
    n. Let the user view the log reports on a remote device such as a phone. (U, F)
    o. Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times. (U, F)
    p. Send a text message to an administrator if an upload/download fails more than its maximum retry number of times. (U, F)

    All the categories include at least one requirement except for implementation requirements, which is empty. You might need to buy new hardware or network bandwidth to support the application, but you're presumably performing uploads and downloads now, so you may already have everything you need. In that case, there are no implementation requirements.

3. Advertising (M) —A phone application typically costs money to install or displays advertising. Currently, the program does neither. It could be modified to display advertising.
   Scoring (S) —Right now you either win or lose. The program could be changed to calculate a score.
   Score keeping (S) —If the program calculates scores, it could keep track of them so that the user can try to beat the previous best score.
   Multiple high scores (S) —If the program tracks high scores, it could be modified to track high scores for multiple users.
   Different fonts (C) —The program could allow users to pick different fonts. (This could be useful if the buttons are too small for users to touch on a phone.)
   Quick win (C)—The program could allow the user to type a guess for the whole word to get extra points. (For example, if you have filled in A_A_E_T_C , you might guess ANAPESTIC all at once.)
   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. Chapter 5 | 371
   Different backgrounds (C) —The program could let the users pick different backgrounds (in addition to the shaded background).
   Different pictures (C) —The program could let the users pick different pictures (in addition to the cartoonish skeleton).
   Random pictures (C) —The program could display random pictures (in addition to the cartoonish skeleton).
   Animated win (C) —When the user wins, the program could display an animation.
   Animated loss (C) —When the user loses, the program could animate the fi nished skeleton (or other picture).
   Report high score (W) —The program could let users report their high scores to a central database so that other users can view them on a web page.
   Animated pictures (W) —The program could display animated pictures that wave, wink, roll their eyes, and so on.
   Word difficulty tracking (W) —The program could track the number of incorrect guesses for each word to determine its difficulty. It would periodically report values to a central database for distribution during later updates.
   Different letter selection mechanisms (W) —The program could allow users to pick letters in different ways. For example, by dragging and dropping letters into specific positions.
   Time limits (W) —The program could display a countdown. Each correct guess would increase the time available.