Andrea Carver

CMSI 402

B.J. Johnson

22 January 2018

*Homework 1*

*Problem 1.1*

Each software engineering project must be able to handle the processes of requirements gathering, high-level design, low-level design, development, testing, deployment, and maintenance.

*Problem 1.2*

Requirements gathering refers to the initial conversations with the client, aiming to clarify their needs.

High-level design involves organizing the software architecture into a few major components and specifying how they interact.

Low-level design involves describing how each of the individual components work.

Development refers to the process of refining the low-level design into actual code.

Testing involves removing as many bugs as reasonably possible, completed by both the author of the code and non-authors.

Deployment involves putting the software into use, including user training, more bug-fixing, and data migration.

Maintenance involves correcting user-found errors and implementing newly requested features.

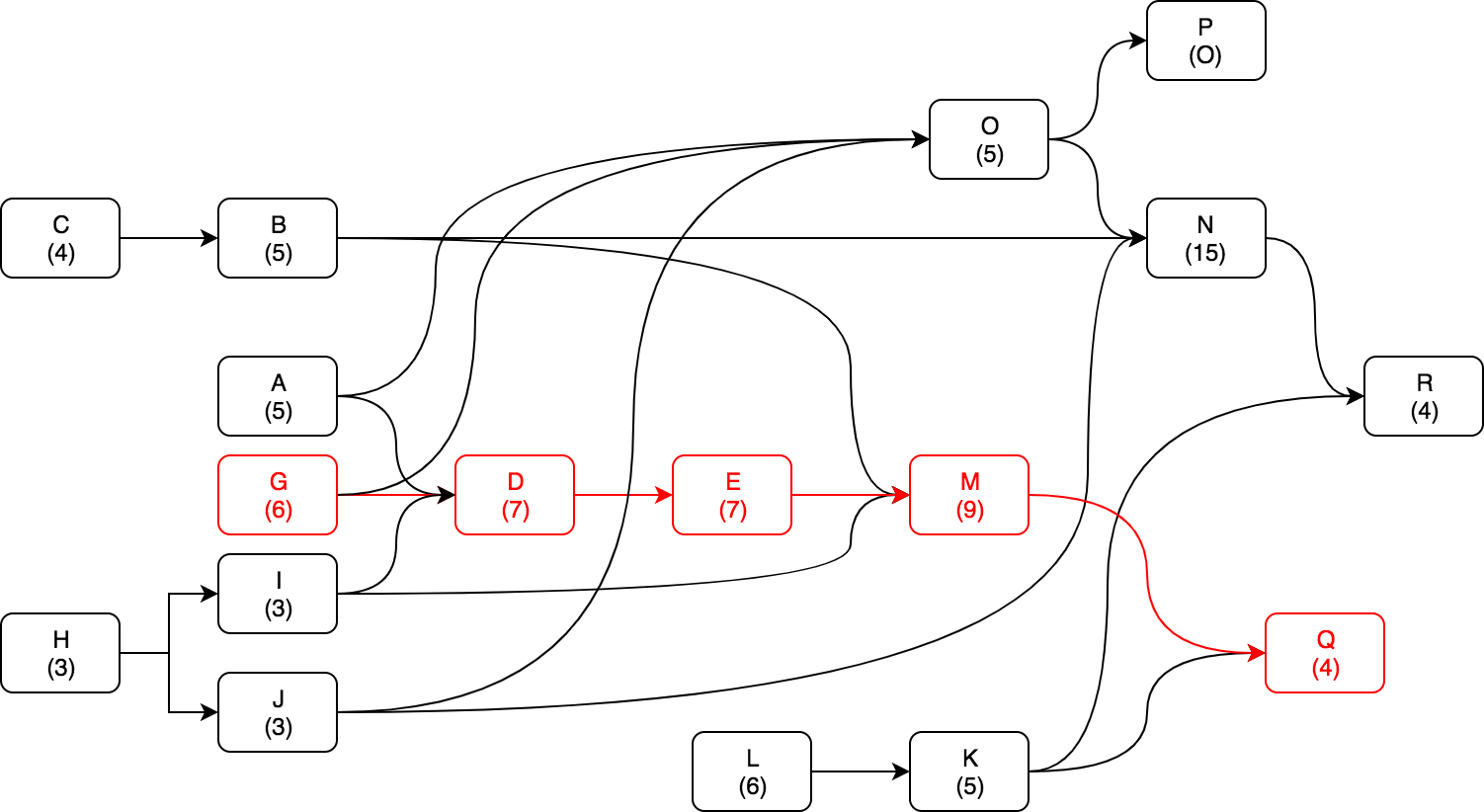
*Problem 2.4*

(No required writing for problem 2.4.)

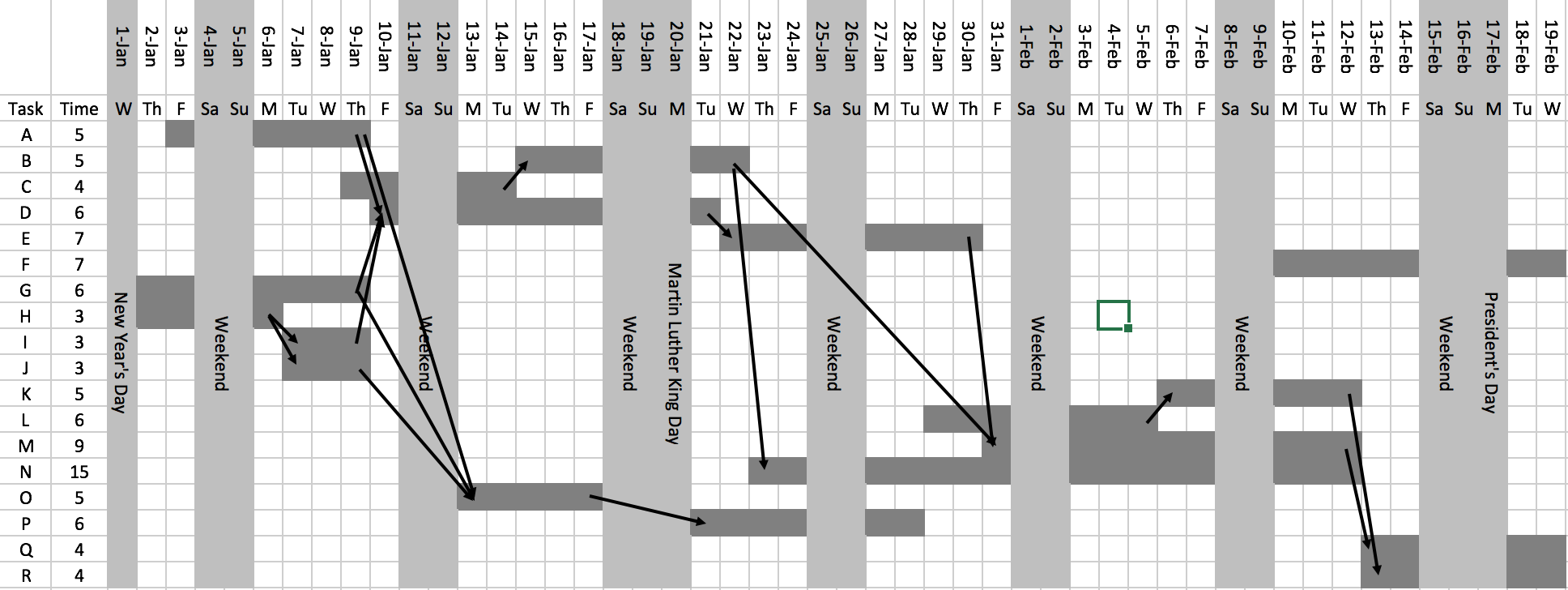
*Problem 2.5*

JBGE means the documentation should be “just barely good enough,” the sentiment that documentation shouldn’t take precedence over the code itself. Many schools of thought exist over how good of an idea this is, since the idea is often taken too far and comments or documentation are outright skipped. Stephen recalls various work experiences where the absence of comments lead to difficulty in understanding code, while writing many comments seemed unnecessary but lead to much greater clarity.

*Problem 3.2*

  
The critical path is highlighted above in red.  
The tasks on the critical path are G, D, E, M, and Q. The critical path length is 33 days.

*Problem 3.4*



*Problem 3.6*

Several strategies can be employed to compensate for unpredictable delays. The allotted time for each task could be expanded by a certain percent, but most people end up using the extra time to complete the original task. A cleverer strategy is to add tasks that represent the delays- this way, original tasks are kept the same length.

*Problem 3.8*

One of the two most critical mistakes while tracking tasks is overly optimistic time estimates. Project managers must gauge each developer’s accuracy in time estimation according to their prior experience with the assigned task(s). Another mistake is to add additional developers to the project to speed its progress. The time to catch them up to speed may exceed to time they save the project.

*Problem 4.1*

Productive requirements are clear, unambiguous, consistent, prioritized, and verifiable.

*Problem 4.3*

1. User / functional
2. User / functional
3. User / functional
4. User / functional
5. Non-functional
6. Non-functional
7. Non-functional
8. Non-functional
9. Non-functional
10. Functional
11. Functional
12. User / functional
13. User / functional
14. User / functional
15. Functional
16. Functional

I did not see any business requirements, as none of the requirements talked about the hopes of the project or any marketing-related goals.

*Problem 4.9*

Must:

* Remove unnecessary “New Game” button and make app start new game upon winning or losing a game.
* Vertically center the UI.

Should:

* Create a title screen that plays for 3-4 seconds before the first game begins, showing the game logo and/or art.
* Put a “?” button in the top right corner. Clicking this button will open a help modal, where instructions can be read.

Could:

* Create word packs, where a user could choose a themed group of words like “American state capitals” or “Aquatic animals”.
* Create a user account-based login, where users can friend-request their friends and send them words to play.

Won’t:

* Make Mr. Bones into a 3D, augmented reality skeleton.
* Release a version of the Mr. Bones hangman game for PC, Xbox, PS4, etc.