1.1 (page 12)

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

Design (high and low level), development, testing, deployment, and maintenance

1.2 (page 12)

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

Design → Decide on what you are gonna make. What platform is it gonna be on? What Resources will you use? And then you will make specific decisions on how the software will operate.

Development → Developers then code the software based on the designs.

Testing → Testers then make sure that the code meets all necessary requirements and does not break down.

Deployment → The product is released for consumers and maintenance starts.

Maintenance → Users find bugs for the devs to fix and work on the product keeps going as the product is still used by consumers.

2.5 (page 26)

What does JBGE stand for and what does it mean?

JBGE stands for “just barely good enough”. It refers to documentation and comments regarding the project. The main idea is that verbose documentation wastes time updating and making progress on the code.

3.2 (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?

Total expected time for each task :

A 5 days

B 9 days

C 4 days

D 12 days

E 19 days

F 7 days

G 6 days

H 3 days

I 6 days

J 6 days

K 17 days

L 12 days

M 28 days

N 26 days

O 11 days

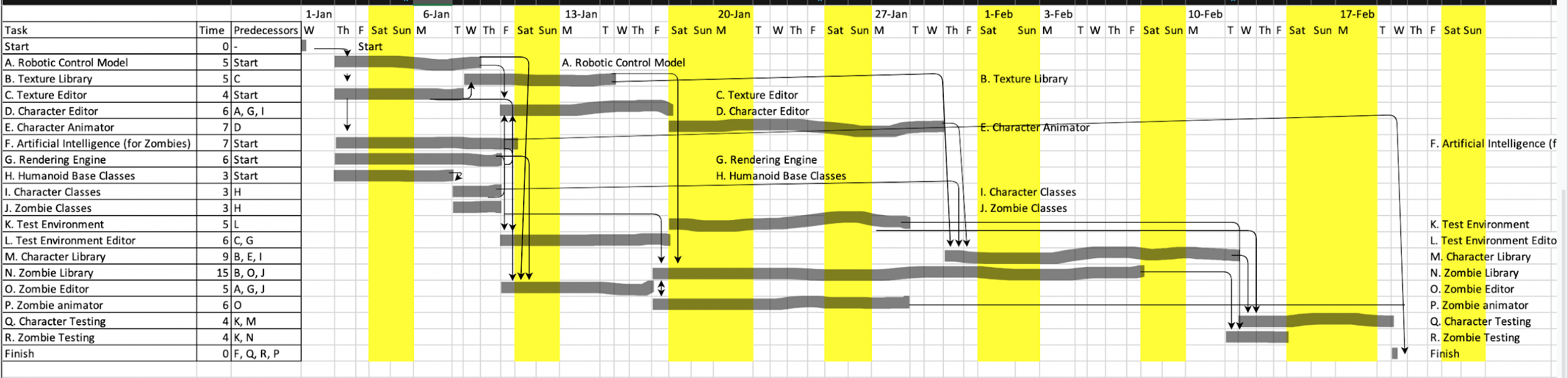
P 17 days

Q 32 days

R 30 days

The critical path for this project is : G-D-E-M-Q. The tasks on critical path are : rendering engine, character editor, character animator, character library, character testing. The total expected duration is 32 days.

3.4 (page 51)



3.6 (page 51)

One can handle these completely unpredictable problems by:

1. Expanding each task time by a certain amount.
2. Add specific tasks to the project to represent lost time.

3.8 (page 51)

The two biggest mistakes that can be made while tracking tasks are:

1. Ignoring the problem and hoping that the time can be made up later.
2. Piling extra developers and assume they can reduce the time to finish it.

4.1 (page 82)

The five characteristics of good requirements are clear/easy to understand, unambiguous, consistent, prioritized, and verifiable.

4.3 (page 82)

1. Business
2. User, Functional
3. User, Functional
4. User, Functional
5. Nonfunctional
6. Nonfunctional
7. Nonfunctional
8. Nonfunctional
9. Nonfunctional
10. Functional
11. Functional
12. User, Functional
13. User, Functional
14. User, Functional
15. User, Functional
16. User, Functional

4.9 (pages 83 & 84)

Must have:

* This phone application could benefit from the implementation of some type of scoring system. For example, a number at the top of the screen representing how many words you have guessed in a row.
* Different skill levels so that all ages of users can play.

Should have:

* A two player function that allows one player to pick the word while the other guesses. This would allow someone to play with their friends or family and promote the game.
* A function that records the top scores giving users something to compete against.

Could have:

* Better animation and or fonts for letters when they are chosen. This would help keep the game interesting and the user engaged.
* Different hangman pictures for the users to unlock once they exceed a certain score.

Won’t have:

* A time limit to keep users from making thoughtful decisions. This would only lower interest in the game and will frustrate players.
* A repetitive selection of words. The game should stay interesting and prevent predictability.