Problem 1.1, Stephens page 13

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

Problem 1.2, Stephens page 13

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

- Requirements Gathering: Determining all of a customers wants and needs for a given project, and compiling them into one concise document for reference.
- High-Level Design: Big picture decisions about what platform to use, what data design to use, interfaces with other system, and high level architectural information about the project.
- Low-Level Design: Takes broken down pieces of high-level design, and provides details and information about how each individual piece should function.
- Development: A process of refining low level designs, writing code for them, and getting rid of bugs within that code.
- Testing: More than just finding bugs, testing includes have the developer and other team members test a section of code, then once complete, integrate that piece into the entire code base to see if there are more errors, and repeat the process until finished.
- Deployment: Finally rolling out the project being worked on, ensure every aspect is put in place, not necessarily only the software alone, but the hardware, etc. as well.
- Maintenance: Becoming aware of, and fixing new bugs that are found by users.
- Wrap-up: Performing a post-mortem to figure out what went well or poorly throughout the lifespan of the project in order to better execute in the future, and finally there is a chance for a break.

Problem 2.4, Stephens page 27

Like Microsoft Word, Google Docs…

Investigation: Compare this process to what you can do with GitHub versions. How are the two tools different? How are they the same?

After clicking on See Version History, Google Docs pulls up a side menu that has each created version and the original blank document to choose from. Each version has a time stamp from when it was created, the creator's name, and the version name. When clicking on each version, the content changes to what was in the document when that version was saved and is highlighted, in my case, in a blue color. After switching to a different version, if that original text or content is also in the next version, it is colored in the automatic text color, and the new, added text is highlighted. The original, empty document is also available to choose from, saved with the timestamp, creator, and the empty word doc. You are also given the option to restore the selected version, which reverts the document to the content and format of that version.

Problem 2.5, Stephens page 27

What does JBGE stand for and what does it mean?

"Just barely good enough," meaning that a task should not be underdone or overdone, and that a middle ground of effort achieves the best overall value to the project. If too much effort is spent on areas that don't need attention, time is wasted that could be allocated to other areas.

Data for Problems 4.2 and 4.4

Table 4.2 [below] summarizes some of the classes and modules you might need (and their unreasonably optimistic expected times) to develop players and zombies for the game. (The program would also need lots of other pieces not listed here to handle other parts of the game.) Use the following table of data for Exercises 4.2 and 4.4.

Task	Time (Days)	Predecessors
A. Robotic control module	5	—
B. Texture library	5	C
C. Texture editor	4	—
D. Character editor	6	A, G, I
E. Character animator	7	D
F. Artificial intelligence (for zombies)	7	—
G. Rendering engine	6	—
H. Humanoid base classes	3	—
I. Character classes	3	H
J. Zombie classes	3	H
K. Test environment	5	L
L. Test environment editor	6	C, G
M. Character library	9	B, E, I
N. Zombie library	15	B, J, O
O. Zombie editor	5	A, G, J
P. Zombie animator	6	O
Q. Character testing	4	K, M
R. Zombie testing	4	K, N

Problem 4.2, Stephens page 78

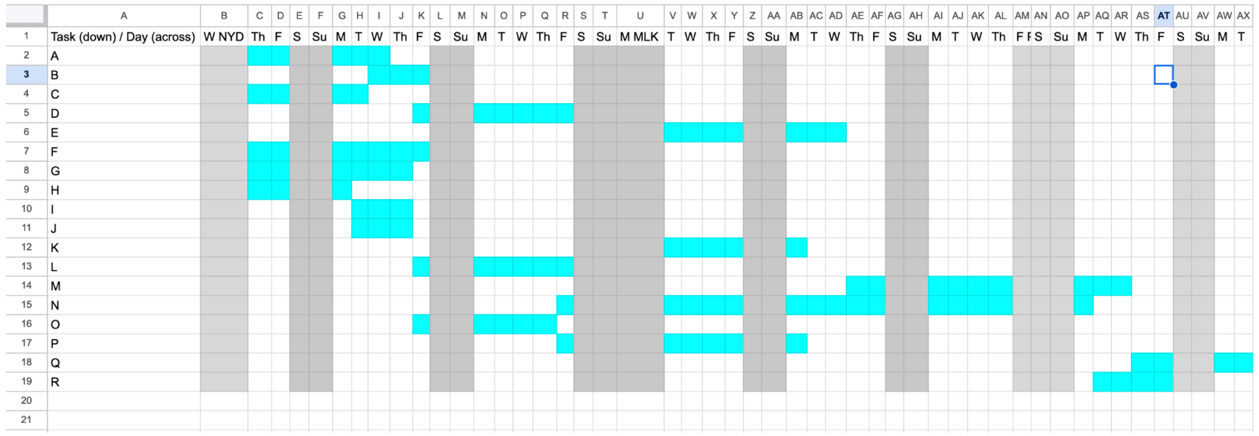
Use critical path methods to find the total expected time from the project's start for each task's completion. Find the critical path. What are the tasks on the critical path? What is the total expected duration of the project in working days?

Critical path: G, D, E, M, Q
Total expected duration of 32 working days

Problem 4.4, Stephens page 78

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, 2024, 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
Alien Overloard Appreciation Day	February 18



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Task (down) / Day (across)	W NYD	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M MLK	T	W	Th	F
2	A		🟦	🟦			🟦	🟦	🟦																
3	B								🟦	🟦	🟦														
4	C		🟦	🟦			🟦	🟦																	
5	D										🟦			🟦	🟦	🟦	🟦	🟦							
6	E																					🟦	🟦	🟦	🟦
7	F		🟦	🟦			🟦	🟦	🟦	🟦	🟦														
8	G		🟦	🟦			🟦	🟦	🟦	🟦															
9	H		🟦	🟦			🟦																		
10	I							🟦	🟦	🟦															
11	J							🟦	🟦	🟦															
12	K																					🟦	🟦	🟦	🟦
13	L										🟦			🟦	🟦	🟦	🟦	🟦							
14	M																								
15	N																	🟦				🟦	🟦	🟦	🟦
16	O										🟦			🟦	🟦	🟦	🟦								
17	P																	🟦				🟦	🟦	🟦	🟦
18	Q																								
19	R																								
20																									
21																									

	A	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX
1	Task (down) / Day (across)	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F F	S	Su	M	T	W	Th	F	S	Su	M	T
2	A																									
3	B																									
4	C																									
5	D																									
6	E			🟦	🟦	🟦																				
7	F																									
8	G																									
9	H																									
10	I																									
11	J																									
12	K			🟦																						
13	L																									
14	M						🟦	🟦			🟦	🟦	🟦	🟦				🟦	🟦	🟦						
15	N			🟦	🟦	🟦	🟦	🟦			🟦	🟦	🟦	🟦				🟦								
16	O																									
17	P			🟦																						
18	Q																				🟦	🟦			🟦	🟦
19	R																		🟦	🟦	🟦	🟦				
20																										
21																										

Problem 4.6, Stephens page 79

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 PCs to the latest smartwatch technology. Or a pandemic, hurricane, trade war, earthquake, alien invasion, and so on could delay the shipment of your new servers. (Not that anything as far-fetched as a pandemic might occur.) Or one of your developers might move to Iceland. How can you handle these sorts of completely unpredictable problems?

A couple ways to handle unpredictable problems would be to expand each tasks' time estimate by some account, to add specific tasks to the project to represent lost time (ex: “vacation time” task), to carefully plan for approvals, order lead times, and setup.

Problem 4.8, Stephens page 79

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

- Ignoring a problem and hoping time can be made up later is a developer is behind on a task, not looking to see if the latest 60% estimate is correct
- To add extra developers to an already failing task, thinking that the task will get done sooner

Problem 5.1, Stephens page 114

List five characteristics of good requirements.

- Clear
- Unambiguous
- Consistent
- Prioritized
- Verifiable

Problem 5.3, Stephens page 114

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.

- a. Allow users to monitor uploads/downloads while away from the office. (User, Business)
- b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password. (User, Functional)
- c. Let the user specify upload/download parameters such a number of retries if there's a problem. (User, Functional)
- d. Let the user select an Internet location, a local file, and a time to perform the upload/download. (User, Functional)
- e. Let the user schedule uploads/downloads at any time. (Nonfunctional)
- f. Allow uploads/downloads to run at any time. (Nonfunctional)
- g. Make uploads/downloads transfer at least 8 Mbps. (Nonfunctional)
- h. Run uploads/downloads sequentially. Two cannot run at the same time. (Nonfunctional)
- i. If an upload/download is scheduled for a time whan another is in progress, it waits until the other one finishes. (Nonfunctional)
- j. Perform schedule uploads/downloads. (Functional)
- k. Keep a log of all attempted uploads/downloads and whether the succeeded. (Nonfunctional)
- l. Let the user empty the log. (User, Functional)
- m. Display reports of upoad/download attempts. (Functional)

- n. Let the user view the log reports on a remote device such as a phone. (User, Functional)
- o. Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times. (User, Functional)
- p. Send a text message to an administrator if an upload/download fails more than it's maximum retury umber of times. (User, Functional)

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

Business requirements lay out the project's high-level goals. They explain what the customer hopes to achieve with the project.
User requirements (which are also called stakeholder requirements by managers who like to use the word "stakeholder"), describe how the project will be used by the eventual end users.
Functional requirements are detailed statements of the project's desired capabilities.
Nonfunctional requirements are statements about the quality of the application's behavior or constraints on how it produces a desired result.
Implementation requirements are temporary features that are needed to transition to using the new system but that will be later discarded

Problem 5.9, Stephens page 115

Figure 5-1 [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."

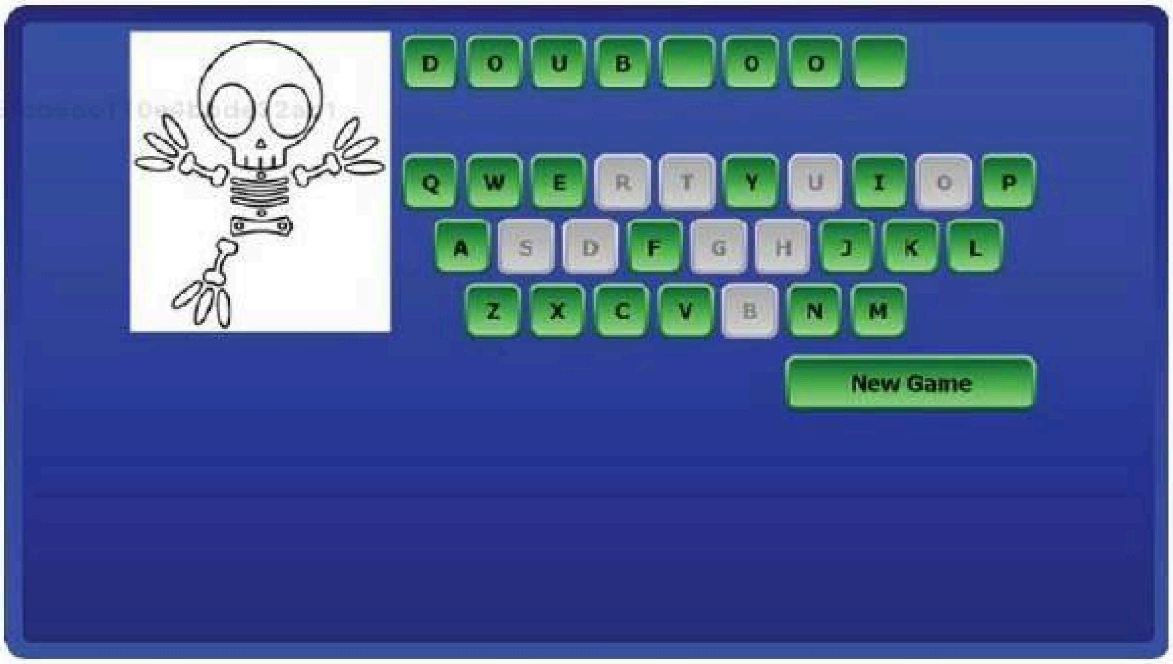


FIGURE 4-1: The Mr. Bones application is a hangman word game for Windows Phone.

Brainstorm this application and see if you can think of ways you might change it. Use the MOSCOW method to prioritize your changes.

M - letter buttons, goal word, skeleton picture

S - new game button, picture filled in with each wrong letter guess, correct letters fill the mystery word, random goal word each new game

C - gray out a letter that has been clicked, win and lose messages

W - color, fluid design / theme, working animations (button clicks, picture fill)