Mark Darling
ISM 4025-01Z
09/4/21

Project 1

Software Proposal v002

PROJECT 1:

I like my visits to the Brevard Zoo. It seemed well run. For this assignment assume that the zoo wants to redo their Zoo Information System that controls most of the zoo operations. Not knowing the full requirements or not knowing much about what the zoo wants to place online, put together a project schedule as defined in chapter 23. It's alright to use your imagination.

Deliverables:

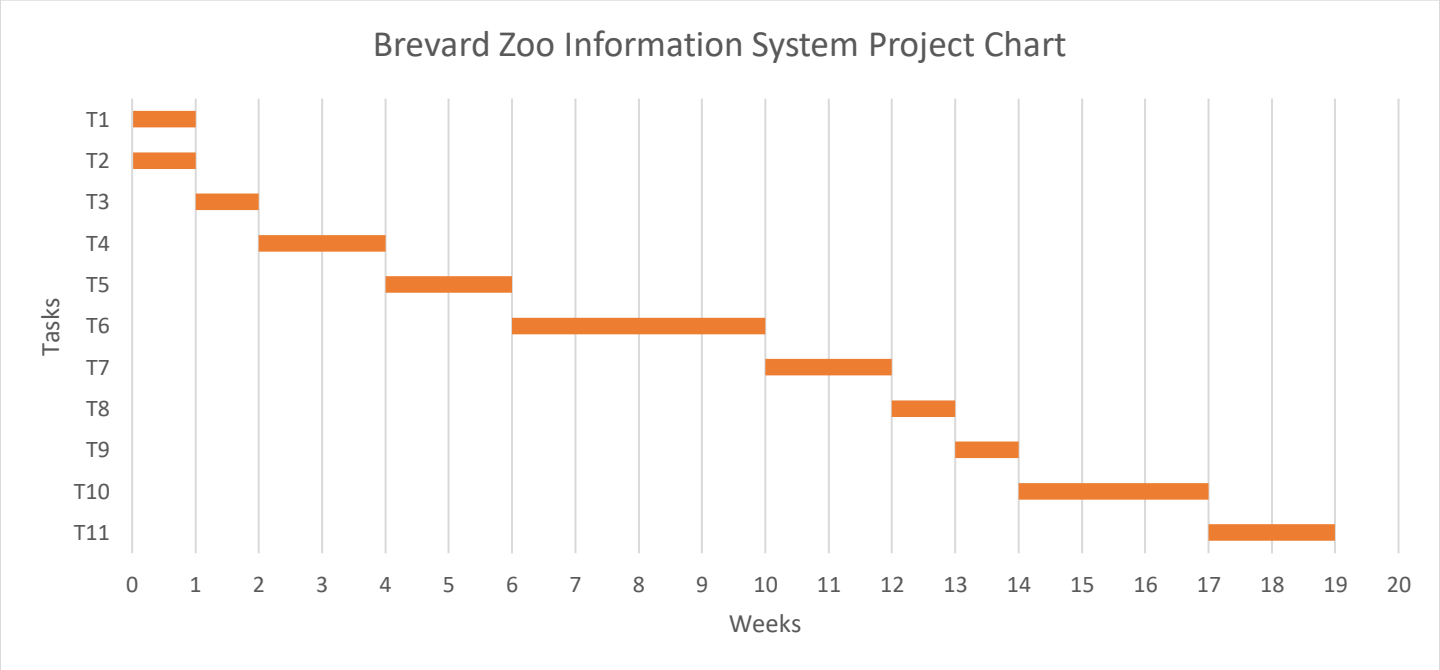
(1) A table with tasks, durations, and dependencies as shown in Figure 23-5: Tasks, duration, and dependencies. Since we are all new to the zoo, we must make a well-educated guess at what the tasks are, how long each will take, and how many people we will need to complete the system. Use the waterfall methodology.

(2) A graphic table with tasks as shown in Figure 23-6. The full system duration will be based on your assumptions from (1) above and task dependencies.

(3) A best guess at the number of lines of source code (SLOC) that will be needed for the Zoo Information System (ZIS). Section 23.5 (Estimation Tech.) from the text may be used as a source. Include all your assumptions that make up your estimate. You may use this SLOC for item (1) above. This is a guess. The final SLOC maybe larger or smaller but for need to need to come up with how much work (requirements, design, coding, testing) is to be done. Go on the web and search for how to estimate a software project.

DELIVERABLES:

Task	Task Description	Effort (person-days)	Duration (days)	Dependencies
T1	Define Project Scope	6	5	
T2	Collect Domain Data	4	5	
T3	Determine Customer Requirements	7	5	T1, T2
T4	Define Project Architecture	15	10	T1, T3 (M1)
T5	Produce Documentation & Specifications	18	10	T1, T3, T4
T6	Code Project Modules	30	20	T4, T5 (M2)
T7	Unit Test Modules	15	10	T5, T6
T8	Test System	9	5	T5, T6, T7 (M3)
T9	Consult Customer w/Pre-Release	7	5	T8
T10	Deploy & Integrate	22	15	T8, T9 (M4)
T11	Maintenance, Debugging, Updates	12	10	T10



SLOC Estimate:

Based on my internet research, the acceptable number of lines of source code per day for a developer / software engineer appears to be anywhere from 10 - 300, with many discussions homing in around 50-100 lines of code per day. Taking the average from that lands at roughly 75 lines of code per day, though many suggest this is still a high figure when the goal is "professional quality, debugged" code. The development language, size of the project, amount of boilerplate, number of tools and/or libraries involved all can drastically change this figure. Regardless, it typically starts out high and tapers down low as the project progresses and more debugging and troubleshooting becomes the greater focus of most developers' days.

Furthermore, the amount of scaffolding or accidental code can vary greatly depending on the specific approach decided on upon implementation. Where one developer may opt for easier-to-read, more verbose language, another developer may prefer to use cryptic, complicated one-liners that achieve the same effect. Even the method used to count the lines of code varies in whether or not to include comments, or non-essential scaffolding type code versus executable statements, etc.

Regarding team size, I initially figured a team size of 12, but after looking into typical / average team sizes a bit for small to average sized projects, I think 4-10 people is more realistic, and landed on a 7-person team as being optimal. Typical team positions include: project manager (1), backend developers (2), frontend developers (2), UI/UX designer (1), and a QA manager (1). I originally estimated duration in days by individual days, but I noticed our book estimated duration in days by the week (multiples of 5) and so revised my estimates to be granular by weeks as well since it also helped make my Gantt Chart easier to graph. Despite wanting to consider some tasks to take less than a week, I will assume the losses in slack time (elaborated below) should hopefully account for some of the differences listed in my chart.

In looking into project scoping further, I saw a recommendation to multiply developer time by 1.5 when estimating time frames since there are meetings, interviews, holidays, communication delays, ramp-up time, and other unexpected tasks that tend to come up in the process. Considering a standard workday is 8 hours, accounting for the slack mentioned above leaves about 6 working hours per day of actual "productive" time, which was considered in re-calculating my person-day estimates based on (duration) day estimates established for the project time frame itself. It was also suggested to re-scope often and to suggest time frames in terms of probabilities rather than best-case-scenarios to remain as realistic as possible.

All said, after revising all of my numbers, I came up with the following considering 4 weeks of code project modules and 2 weeks of unit test modules being developed, though I imagine the final figure rests somewhere between the two SLOC estimates calculated below, especially because as the number of people on the project is increased, so too is the total amount of slack time lost amongst them collectively:

TOTAL PROJECT LENGTH	19 weeks
SLOC ESTIMATE (*days*)	30 days * 5 people * 75 lines/day/person = 11,250 LoC
SLOC ESTIMATE (*person-days*)	45 person-days * 1 person * 75 lines/day/person = 3,375 LoC
TEAM SIZE	7 people
CODERS	4-5 people (depending whether QA manager counts)