SEPM Presentation Transcript

Please use the following link to view the presentation recording which includes this slideshow, cost estimator application, and code review (19:57): Final SEPM Team Project.mp4

Haaris Mian
Hello, everyone, thank you so much for joining us today we are Team 3 of the software engineering project management class, consisting of myself, Haaris Mian, Bradley Graham, Tarek Goda,, and Michael Sammueller. And today we'll be talking about our response to the EDC Complaints given our initial designs for the Synputer project.

to kick us off. Let's go over a bit of an agenda. First we'll start off with our project assumptions, namely, the triple constraints. Brad will then talk about the EDC complaints to our initial design as well as our formal and official response. We'll follow that up with a project plan as well as milestones that we hope to achieve during the course of the project. I'll follow it up with project risks that we believe might exist to the second iteration. Then Tarek will demonstrate our cost estimator application. And finally, Michael will close the presentation off going over the code base for our cost estimator application, demonstrating its robust commented and high quality code. Take us away, Brad

Bradley Graham
Our project assumptions have been categorized by project constraints identified in the 1950s for general projects.

The project length we believe to be 13 months long, and we believe will be using an iterative iterative software development lifecycle include design, build, test, and deliver phases as talked about by Larman in 2003. We'll also be using agile principles from the agile manifesto in 2001 and more specifically, we'll use a scrum framework which was developed by Rising & Janoff in 2000.

The other important thing related to time are the actual components, specific timeframes, so different components have slightly different design, manufacture, and testing times, so we also take that into account as well using information provided to us.

As for the cost of the project we assume the budget is 500,000 pounds. We have a market price of 400 pounds and the possibility of a second system marketed for 440 pounds.

For system costs. We look at three resources available to the company. The price of system components, licenses and the staff wages. And we use estimation techniques, including COCOMO by Boem in 1981. And we also use our in-house, task-based prediction application.

We are assuming that we have the tools available for us to actually build these systems. And we're assuming there will be risk costs as well

SEPM Presentation Transcript

For the project scope. Our main stakeholders are EDC, the company we're creating 2,000 systems for, and our own internal stakeholders. The project vision includes the EDC specification to build a business application. We've got some information from our internal development research team regarding what is easy to develop, and, what is more difficult, and we've also got some marketing research results which tell us that we probably should be building a gaming application that's upgradeable for a second system. However, we will not be building this second system as explained later.

So just to recap where we are, we sent a design to EDC, and they replied, saying that they were disappointed in the design, and they wanted us to meet at least 80% of a new set of requirements they set. Those requirements specifically are that our system has a board that supports GUI and a mouse. It has an industry standard removable drive. It has 512 GB of RAM. It has a minimum, 2 RS 422/485 standard serial ports. It has SCSI expansion, capability, an industry standard operating system, and at least the 68 K CPU.

We believe we can actually meet all of these requirements, although we did prioritize them in terms of how flexible the requirements are. So in order to address these requirements for the industry standard operate system. We go with MCCOS, for the SCSI expansion capability, we use a pro expansion board which has an SCSI capability and a bunch of ports and sockets for other things which we would meet in this system. For the 512 KB of RAM we go for 2 256 KB RAM chips, for the industry standard removable drive, we go with 3.5“ inch floppy storage for the board to support GUI and Mouse. We make sure that we've got a GDISP XVX And to have 2 Rs. 422/485 standard serial ports. We also go with an OP-J and an IOP-J-2 input output chips.

So our overall system then includes a 68 K CPU pro expansion boards, 40 KB of ROM which includes an 8 kb. Chip and a 32 kB chip, and then 512 kb RAM, i8042, an XVX GDISP and OP-J. And IOP-J-2, input- output chips, resistors and caps, a desktop case and 3.5” inch floppy drive. And the software is mainly the MCC OS and a telebacic programming language support. This system costs less than 250 pounds to create, including the prices of the components and the human resources. So that allows us to build 2,000 of these using our 500,000 pounds budget to meet Edc's requirements.

We can actually create this system in about 56 worker weeks. The main milestones rated here are the 24 worker weeks to build the hardware for the first system, and it will take us roughly 16 weeks of design work for the software and an additional 32 weeks for coding the software. So because we can begin the design during the hardware development phase in total, we end up with 56 workers weeks. That's not to say the project will take 56 weeks long, because we'll be working with 2 teams. One hardware team and one software team in parallel, and both teams will have 3 people, although our budget only allows 2 workers to be working at the same time in each team, so they will have to have some sort of rotor, which they can decide for the software priorities we're assuming there's going to be a 2, 1, 3 ratio for design, manufacturing and testing, which coincidentally agrees with Brooks rule of thumb from 1975, which is basically saying we should be generally performing more testing and designing for software, and our figures actually

agree with that. However, for the hardware priorities we predict a 1, 2, 3, ratio for design, manufacturing and testing time.

The actual project schedule consists of either 6 week long governance cycles and those governance cycles are specifically relevant to project analyst and the project manager. So the project analyst will be available at the beginning of each cycle to check the project's state, and just to generally track the project and risk tracking and to update the project if it needs to have some major changes, and they will also work with project resourcing. The project manager will be available fortnightly through those governance cycles and perform in smaller less analytic roles: but will be there available for the teams. So each 6 week long governance cycle then will be made up of 2 or 3 sprint sprints and the flexibility there is purely for the developers and architects' own flexibility in deciding how to go about building these systems, the idea, then, having the shorter sprints for their flexibility, but these larger governance cycles for giant shifts, if that is necessary, because there's always a risk that that will happen.

The main milestones of the project are by week 12. We expect half the first machine to be hardware complete by week 28 we expect the first machine to be software complete, and by week 32 we expect 2,000 systems to be hardware complete. And by week 48, we expect 2,000 systems to be software complete. So then by week, 56, we will have 2,000 machines that we can deliver to EDC, which gives us about an 8 week buffer to account for anything unexpected that happens during the development.

Another key point here is that actually, the hardware and software effort can be refocused on designing gaming systems later during the project. But the new gaming system is not going to be ready by the first year. It's definitely something on our radar, but we want to make sure that this business system is as best as it can be. We don't want to increase the risk unnecessarily, as we believe we can actually profit off this system, because once we've gone through the initial 48 weeks to create the 2,000 systems that we're giving away from our budget. Our costs of developing new systems will be much cheaper, and we had to profit much more greatly from them. We believe that the advertising from the EDC And the contacts we gain from them will help us propel increase our momentum for future sales. We can also start getting universities involved with our newer systems as a form of advertising as well. And once we've developed another budget or gained additional budget from someone else. We would then look into the gaming system.

Haaris Mian
Thanks for that detailed overview of the project and our plan, Brad. Now let's quickly go over some project risks. No project, despite how well planned, is without risk. Now these are listed 4 main risks that we believe might exist to our project as well as our mitigation strategy, including an estimation of the likelihood and impact pre and post mitigation.

Firstly, we believe that the possibility of project overruns in time, scope and cost, and our mitigation strategy is to implement a robust project management system and such as we have

done using agile and scrum to allow flexibility and regular reassessment of our timeline and budget utilizing iterative development, we can make adjustments as needed.

Just as Brett mentioned, there is a risk of insufficient staffing or skills gaps. Given the innovative nature of our project, we believe that this is not very likely or impactful. However, we've implemented a mitigation strategy where we will plan for adequate training and development of current staff, and, if necessary, hire and rotate skills as necessary.

We also believe what got us into this very situation. In the first place, with EDC Which is for communication among team members or stakeholders. Now, it's our goal to move forward with EDC by establishing clear communication channels and have regular update meetings as well as use collaboration tools to enhance interaction and maintain transparency with our stakeholders. So we don't arrive at another situation in which the client is dissatisfied with the project. We want to make sure that everyone is on the same page during the entirety of the project.

Finally, not very likely given the high quality team and plan we have in place. But the risk of quality issues in the final product. While this isn't likely, we believe it would be very impactful. As a result, we've implemented a rigorous quality assurance process conducting regular testing phases throughout the Development cycle, thanks to the agile methodology where we will address issues promptly and immediately.

Now I pass it off to Tarek, who'll show off the cost estimation application which will help in the project planning and iterative iterative phases of our Synputer project.

Tarek Goda
Hello, and welcome to this section of the presentation, where I'll be demonstrating the cost calculator application.

If you have problems navigating the application, please take a look at the README File which contains all the usage and instructions to use the cost estimation tool we provided. After you download the file from Github, extract it you will see the file structure up here. All you have to know is that the data file contains all the JSON data contains also the keys and values of the software and hardware components seen here.

if you'd like to run this application, either use the user interface. The files from the application should pull the cli or command line interface, using the terminal or through the GUI, also known as the graphical user interface.

First, I would like to navigate through the terminal, as you can see, here are the important modules, including OS, JSON, which will be used to read tables, and, of course, calculations to get the project class and find the calculations of the file.

SEPM Presentation Transcript

The following class contains functions that will allow the user to navigate through the terminal window. These are basic functions, including Pd, or view data function that will enter into data and provide over the listed dictionaries and lists and view them in a table form using this method by pandas.

I'll quickly navigate through the terminal window all you have to do is simply run this file and then locate the JSON file after reading through the directory you're following the Json file, where names will appear. All you have to do is copy them and paste them in the terminal window. Make sure the Json file is in the data directory or file structure.

After loading the JSON files you can type pd to see the hardware components and software and resources listed on the window and to do a quick calculation on these. All you have to do is type CLC. As you can see the totals are visible in the terminal window.

If you would like to navigate to this application through the GUI, then run the gui.py file

So this window has calling functionality where you can upload, make new json edit, add new keys and values, or delete, or even save just a new file. And of course, to update the costs. All you have to do is click the double button. Have it load this data and open it using the tree viewer white in general, review these data from the JSON.

If you'd like to quickly calculate this data all you have to do is hit the calculate button in the bottom corner.

If you'd like to edit any of the keys and values you just have to click on the key or value in the main window, and click in the field and edit the value and hit the edit button. You can see the budget change. If, for example, you like the desktop vision feature, you can, for example, change the price here.

Edit this function, calculate, and you can see the cost change corresponds to the values we have changed in the value box.

If you'd like to delete any of the rows, you can simply click on the row and hit Delete

If you would like to build a new Json file new data, all you have to do is click the file menu and click new and this will allow you to save as a JSON, and will allow you to add new rows and listed dictionaries. So, for example, our test one with a value of one thousand. and then add, as you can see, the values added in this form are viewed in the tree viewer. You can also add values and keys within the main data. So, for example, click, add. As you can see, we did this here and can also click, add on the time it is here, and so on, and it is saved.

You can also test the functionality by uploading the old JSON file.

If you would like to select or deselect. You simply hit the right or left button so. by selecting the row, the correspondent values and keys will appear. can be selected with a right click. and that will be the administration of the GUI interface.

Michael Sammueller
For our project, we decided to follow an object-oriented paradigm. We created four classes:

1. The Project Estimator class
2. The Software Component class
3. The Hardware Component class
4. The StaffMember class

The Project Estimator is the main class, which includes empty dictionaries for instances of each of the three other classes in its constructor. It is the main class of our program because it hosts the functionality to calculate and estimate project costs. Whenever the user starts the program, an instance of the ProjectEstimator is created, together with those three empty dictionaries.

The Software Component class, as well as the Hardware Component class, represent the individual components necessary to make up a computer system. Using these classes, we can define all necessary details such as types, costs, and man-weeks required. All this information can then be used by our calculation algorithms to estimate project cost.

The Staff Member class, similarly to the component classes, allows us to define staff members in great detail, including their titles, types, costs, and work capacity, which we have capped at 260 days to match our project deadline.

When the user selects and uploads a JSON file into our software, the “read_json_data()” method of the ProjectEstimator class is triggered.
This method iterates over the JSON data returned from the GUI, creates instances of each hardware component, software component, and staff member, and adds those to the relevant dictionaries. Doing so allows us to retrieve all the necessary information to perform our calculations.

Not only does the Estimator calculate a grand total, but it is capable of calculating each aspect of a software engineering project. This includes:

1. Individual costs like: design cost, manufacturing cost, coding cost, and testing cost
2. Grad Totals like: total staff cost, total cost for 2000 systems, cost per system, as well as an estimation of cost using COCOMO.

Using COCOMO allows us to roughly estimate the total project cost from a software perspective. As most algorithmic representations of estimation methods focus on software

rather than hardware, we use this as a rough guideline to estimate software cost, including the staff cost to produce said software.

In order to use the COCOMO algorithm, we have made some assumptions as to the capabilities of the software team, the level of experience of individual staff members, as well as the efficiency of their processes. These assumptions have been clearly mentioned within the code.

To allow for re-use and customisation of our program, we can specify which of the modes defined in the COCOMO model fit best to a project, allowing the end-user to get an estimation most fitting to their project type.

When the user clicks the “Calculate” button on the GUI, or issues the “Calculate” command in the CLI, all of our calculation and estimation methods are triggered to produce the final results seen in the terminal or on the graphical user interface.

In order to perform these calculations, we require two types of methods.

1. Methods that iterate over the costs and counts of each component and return a grand total for the component type
2. Methods that iterate over all components and match these components to the relevant staff members.

The latter is far more complex as it involves several steps to produce an accurate result. By matching staff “skills” to their counterparts in the “required_skills” list of each component, we assign staff members ensuring that the cheapest staff members are used first, and that their work capacity is not exceeded.

Since Synful has a relatively small team, there is a need to hire third party agency staff, which are far more expensive than the in-house staff. Using our methods, we ensure that the in-house staff is assigned first, until it is absolutely necessary to assign agency staff, to save costs. This means, that even if a large team of staff members are assigned to the project, as part of the JSON file, our program ensures that staff are assigned as efficiently and cost-effective as possible.

TESTING

To ensure functionality and a high standard of code quality, we tested our code using both Python’s inbuilt “unittest” library, as well as “pylint”. As the size of our code grew and more functionality was added, we ensured that features such as instance creation and mathematical calculations were working correctly. We also made sure that our code adheres to the “Zen of Python” and upholds a high standard of quality, achieving an overall pylint score of 8.5 out of 10. More information about individual features of our program, as well as which tests were performed, can be found in the README file.

Haaris Mian
And that concludes our presentation. Thank you so much for taking the time today to watch.

References

Atkinson, R. (1999) Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *Internattional Journal of Project Management* 17(6): 337-342. DOI: https://doi.org/10.1016/S0263-7863(98)00069-6

Larman, C. & Basili, V. (2003) Iterative and Incremental Development: A Brief History. *Computer 36*: 47-56. DOI: https://doi.ieeecomputersociety.org/10.1109/MC.2003.1204375

Rising, L. & Janoff, N. (2000) The Scrum software development process for small teams. *IEEE Software* 17(4): 26-32. DOI: https://doi.org/10.1109/52.854065

Beck, K., Beedle, M., Bennekum, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001) Agile Manifesto. Available from: https://agilemanifesto.org/principles.html

Finkelstein, M. (2008) *Failure Rate Modelling for Reliability and Risk*. Springer. Available from: https://doi.org/10.1007%2F978-1-84800-986-8_1

Boem, B. (1981) *Software Engineering Economics*. 1st ed. Prentice Hall.

Brooks, F. (1975) *The Mythical Man-Month*. 1st ed. Addison-Wesley.