CPSC:480 Software Engineering Project 4

Scenario

The only real fiction in this scenario is that anything will move forward in 30 days.

In the previous sprint (projects 2 and 3), you made progress toward a minimal prototype of your software project. However, your team has just been acquired by my company "EdgeCase Solutions, LLC" for further development next semester. As part of the acquisition, I will be restructuring the organization for efficiency. This means that in 30 days, all six teams will be merged to work on just one project for another five months, abandoning the other software projects. There may also be layoffs of Twitter-like proportions.

This announcement interrupts your planned sprint and release schedule. The new owner would like you to focus your development activities on a presentation of your project for the final sprint before deciding whether to move forward., This will be given at a meeting scheduled for November 28, including the new organization owner (me) and everyone on the other teams, who might be joining your project. You will also submit a written report defending the case that your project and your team would be likely to succeed at the new company.

It is well understood that you just started on this software development project and probably are not nearing completion of a releasable product. You will need to be upfront about the state of your software – what it does and does not do – but the new owner is interested in the bigger picture. A successful software team should be able to demonstrate that they have:

- A well-defined and appropriate software development process and evidence that they've followed it
- Thorough product testing and documentation
- Quality as a priority consideration throughout the whole project
- The agility to adapt to changing requirement priorities
- A vision for what the software could do in the next six months and beyond
- Compelling arguments for the likely success of their project
- Dedication from every member, positive team dynamics, and mutual trust
- Honest assessment of strengths & weaknesses and a plan to improve
- The ability to convince other engineers of the project's potential

Part 1: Incorporate Requirements Change

The set of project requirements will not change as a result of this takeover. However, the prioritization of them should change from ordering tasks by importance to the eventual end user, and instead prioritize based on completing what you can implement by Tuesday, November 22, for the best show on Nov 28.

To estimate what you can reasonably complete in this time, refer to your initial estimates and velocity calculations from the previous sprint. Meet (in-person or virtually) and discuss what went well and what didn't with accomplishments vs expectations from that sprint, and what you can do to improve your development process. Identify the requirements that you plan to complete. Revise or complete task lists and estimates for items selected for this sprint where needed, and assign these tasks to be completed in part 2. Identify the major technical and logistical project risks that are most likely to prevent meeting the selected requirements, and determine how you can minimize this risk. It is not required that programming work be distributed evenly or even that everyone contributes to the codebase. A designated team member should take thorough notes from this sprint retrospective and planning meeting.

Part 2: Minimal prototype completion

Assigned developers should complete development tasks and produce a build of their demo prototype with a target "pencils down" date of Tuesday, Nov 22, 11:59 PM. After this point, the demo prototype should be considered code-complete and the only changes from this point on should be bug fixes, minor adjustments, code quality improvements, or tests, and you will need to follow a change control process to make these changes. Ensure your process accounts for code review, automated testing, and manual testing for regressions and new issues. For checkpoint 1, you will need to publish the code-complete build as a GitHub release by the target date (Tuesday, Nov 22).

Part 3: Additional Materials

In addition to the initial prototype of your software product, you will need to provide additional artifacts relevant to your project. This includes:

1. A summary of the code quality and technical debt in your project.
2. A description of how you've followed a software engineering process effectively or cases where you haven't followed the process.
3. A project plan and timeline for how your product could be launched with another semester or less of work and what resources would be required.
4. An overview of the codebase and repository aimed at engineers who might be joining your project.
5. Major risks identified for the project over the course of the next semester. Use a risk assessment form or follow the format at the end of Risk lecture.
6. Part 1 meeting notes
7. Change control process documentation from part 2
8. Product documentation describing operation of your product.
9. Technical documentation on building, testing, and installing the product.
10. Identification of the function in your code with the highest cyclomatic complexity, what the value is, and how it could be refactored or why it's okay as is.
11. A description of how your project uses or could use automation in the software development process.

Additionally, your GitHub repository should have the following:

1. Work items with task lists, original estimates, and comments with time taken to implement and test.
2. Bug reports for all known issues in your product
3. Clean commit history and repository organization
4. Revision history of all artifacts
5. Pull requests and code reviews for changes

Submit the report by compiling this information into a project synopsis and posting it as a pdf in a release in your GitHub repository. For checkpoint 2, you will need to create this release by Sunday, Nov 27, 11:59 PM.

Part 4: Presentation

For selecting a project to move forward with, the main event on the new owner's mind is the meeting for project presentations on Monday, Nov 28. Your presentation should aim to do the following:

- Inform the audience about your development process, product design, and pre-existing implementation work, and what would be done next semester
- Convince the new owner that your product and plan to build it would have a good chance of success if it were to continue development.
- Convince the new owner that your mastery of the software development process would make you good candidates to stay with the company.
- Convince your peers that your project would be worth joining and staying on for the semester after the teams merge.

You will have 10 minutes timed for the presentation, with points deducted for running over, and a hard cutoff at 11 minutes. You should have your presentation uploaded to your repository as a .pptx and a .pdf by 3:00 PM on Monday, Nov 28, and you should create a GitHub release by this time with the exact code you will demo in your presentation. Your demo should be ready to present on a device of your choosing with HDMI output, and a demo should optimally take around 1-2 minutes of your presentation time. You do not need to have everyone on the team speak. Two presenters is actually ideal – one for the slides and one for the software demo.

Part 5: Final Release

Review the feedback you get from the new owner and potential teammates and determine what changes you might need to make as a result. Add the change proposals into your repository. You do not need to actually make all the changes you propose, but this will also be your final opportunity to make improvements to your product and accompanying artifacts. Create the final GitHub release by Friday, Dec 2, 11:59 PM. Be sure to follow your change control process and track your changes. A new owner would reasonably consider the team's incorporation of stakeholder feedback and continued development efforts in their review.

Part 6: Individual Report

As a final component of the new owner's project evaluation, you must complete an individual assessment of your project and make your own recommendations for how the new organization should proceed. Write your own report defending your project, your team's performance, and your own individual contributions. Include your conclusions on the best course of action for the combined organization selecting from the six projects for continued development.

Describe your own contributions with the possibility in mind that a new owner might choose to keep on proven individual high-performing engineers regardless of the success of the rest of their team, project, or product. Describing contributions is not a competition; your accomplishments don't take away from anyone else's and vice versa. Ideally you should help each other best portray your strengths to new management. If the new owner believed it was worthwhile to keep everyone on, they would do so.

Submission

The deliverables are the meeting notes from part 1 (submitted in the part 3 report), the work item changes for part 1 (repository issues), the change control process in part 2 (also submitted in the part 3 report), the prototype published for checkpoint 2 (GitHub release), the report in part 3 (GitHub release), the repository contents in part 3, the presentation and demo in part 4 (live + presentation checked in), the change proposals and final product/artifact changes in part 5 (GitHub release), and the report in part 6 (Brightspace). Submit any changes to the repository through GitHub using your change control process, and submit the part 6 individual report and post-project survey through Brightspace. The software demo is the only component that will be assessed solely in class.

The deadline for final submission of all materials is Friday, December 2, at 11:59 PM. **No** additional work will be considered after this point. You will receive 1% extra credit if all of *your* repo changes and Brightspace materials are submitted by 6 PM on the due date (Friday, Dec 2, once again). You will receive 2% extra credit if all of the *team's* repository changes and all of *your* individual materials are submitted by Thursday, Dec 1, 11:59 PM. Any teammates' commits to the source repository after this time will forfeit the 2% extra credit, but the 1% extra credit would still be available if you meet the terms for it.

Grades

Grades will be 20% individual and 80% shared, broken down evenly as follows:

- 10% – Part 1 meeting notes and artifacts
- 10% – Part 2 implementation
- 10% – Checkpoint 1 requirements met
- 10% – Part 3 materials
- 10% – Checkpoint 2 requirements met
- 10% – Part 4 project presentation
- 10% – Part 4 prototype demonstration
- 10% – Part 5 final submission
- 10% – Part 6 report (individual)
- 10% – Effective defense of your project and contributions (graded individually, but team presentation, materials, and other resources can factor in to your defense)

Members of the top 3 teams “selected to stay with the new company as engineers” will each receive 2% extra credit on the project. The one project “selected for the new company to move forward with using the combined team” will receive recognition, but no additional grade credit. Individual extra credit or penalties based on individual contributions may also be assigned based on peer and instructor assessments.

Commentary

Note that it's possible in this scenario for the project to be selected from a team that didn't make top three. This would happen for example if one team had a clearly superior product design and prototype but other teams demonstrated more effective software engineering and project management skills. In the real world, of course, that would be up to the new owner, and they could choose to do that. But six-way mergers followed by immediate, deep layoffs and project cancellations would be a terrible business strategy, and employee survivorship being determined by presentations in front of each other would be horrifying for all involved. I am not aware of a case where any similar scenario has occurred, but framing the assignment this way serves to illustrate several engineering concepts.

In particular, the scale and type of this requirements change (reprioritizing existing requirements between sprints and establishing milestones for the upcoming months, while taking previous work into account) would be similar to those you might encounter on a real project and that you would be expected to accommodate. Stakeholder feedback meetings are also a regular part of software engineering work, ranging broadly from relatively inconsequential meetings showing one change to the team at a sprint review, to public or semi-private presentations (e.g. academic conferences), to major pitches for customers or investors. Audits of software development organizations are also routine (see SOC II Type 2), specifically looking for evidence that the team has established appropriate practices and processes and consistently follow them. Employee performance reviews are yet another routine occurrence; they're *usually* not competitive ("stack ranking" is out of favor), and being a contributing member of a high-performing team will usually earn you higher marks than comparable work on a lower-performing team.

More practically for the course, this scenario framing establishes expectations and intended approach for the project work. What the software needs to do now is "enough for a stakeholder review meeting". What you need to plan for is bringing the project to maturity over another semester. Following best practices in project management and software engineering processes while maintaining quality throughout the projects will improve the defense you can make. And arguing for your prospects of success puts the responsibility on you to identify key contributions and highlights to consider when assigning grades.