CPSC544 Advance Software Process

Group Project

Group Members:

Alvin Godoy

Derek Hirahara

Michael Meier

Peter Tran

Vorleak Yek

California State University, Fullerton

Professor: Jason Choi

Executive Summary

Note: The executive summary of an evaluation report is a shortened version of a full report. It highlights evaluation findings and recommendations, but may also include a brief overview of the evaluation purpose, key evaluation questions, and the research options used.

There are three critical issues that Collabo is currently facing in the business-to-feature phase. The first issue is that features are not properly reviewed and prioritized. The second issue is that features are disorganized and not having enough requirements. The third issue is that there is no change control management so that features and feature requirements can be easily changed anytime. To resolve these issues, we recommend the team create a detailed product plan/roadmap that includes a list of major features with high-level release dates. Having a clear roadmap will help the product owners to prioritize Collabo's features before a big release. We also suggest that the product owners organize each story based on an epic for the project. The teams need to establish stricter change management after the initial baseline has been signed off and conduct impact analysis for any proposed changes.

There are two main issues with the testing phase. One issue is that the quality engineering teams came later in the software development life cycle, which causes delays in releases. The other issue is that the automation effort was not successfully built due to resource constraints and the lack of sponsorship from upper management. Our recommendation is that Collabo should start the testing phase earlier in the software development life cycle to avoid late critical issues found. Scrum Agile framework might be a good fit since it involves testing before codes are merged to the baseline. This will give Software Testers enough time to find critical issues and allow the developers to fix them before building more code on top of those bugs. In addition, we recommend that upper managers consider switching to DevOps if possible since it is an efficient way to boost automation to the next level.

In terms of integration and release, there are two main issues. One of them is the nature in which the process is performed during deployment. The leads for each development team must manually take down, upload, and install the build of the new release from the server before starting the service and testing for any defects. This current process poses risks for build inconsistencies and errors due to human interaction with the system. The other issue Collabo deals with stems out of the first one in which it requires downtime for every deployment of a new release. The downtime, even though minimized, still has that window period when users are not able to access the software therefore causing potential loss in productivity, and in turn, revenue for the businesses using the product. One of the recommendations our team came up with is to first automate the process. Having the Infrastructure as Code (IaC) eliminates risks of errors due to manual processing. It also enables maintenance of the configuration version history, if stored in a version control tool, and facilitates standardization. With the automated setup, it makes it easy to build redundant servers. Having multiple copies of the builds, downtime can be eliminated allowing users to continuously use the product even during maintenance.

The current state of Collabo's defect tracking is an underdeveloped system that is not equipped to handle the current magnitude of reported issues. Due to the system still being in its early stages, they do not have a good way to classify and organize the reported issues. This leads to many repeated issues, poorly classified issues, and untraceable issues. All of this combined is causing the backlog to grow uncontrollably.

Introduction

An analysis of the current state of the business highlighted many key areas where improvements could be implemented. The analysis determined that there were issues in six key areas: business to feature, planning, development, testing, integration and release, and defect tracking. The business-to-feature phase was suffering due to a lack of key feature prioritization. The planning phase is not implementing and adhering to any modern development strategies. The development team is heading in the right direction but lacks any consistency and standards within their code. The testing phase, one of the worst, has been set back since the beginning due to not having a dedicated quality and testing team. The integration and release are going poorly because of a waterfall effect from the other phases. Lastly, defect tracking has had a variety of issues stemming from not having a comprehensive defect management system. It was made clear that targeting these areas can result in major improvements to the efficiency and success of Collabo.

Collabo is a promising project that has great potential. However, it is in a critical period where it needs to progress out of the early stages of software development and start to implement repeatable development strategies. The company is progressing too fast to stay in phase 1 of the development process and will begin to see more and more issues if it does improve.

Analysis and Recommendations

Business to Feature Phase Analysis and Recommendation:

Current State Analysis:

There are three main issues that Collabo is currently facing for this phase. The first issue is that product owners do not properly review and prioritize Collabo's features. The second issue is that many of the features are disorganized and do not have enough requirements for developers to understand during the implementation phase. The third issue is that product owners are changing features and feature requirements without any discussion with the team or documenting the changes.

Recommendations:

We recommend Collabo teams create a detailed product plan/roadmap that includes a list of major features with high-level release dates. These features should have been confirmed by the product champions that they are the desired features that the customers want. Having a clear

roadmap will help the product owners to prioritize Collabo's features before a big release. In addition, we suggest that product owners review the product strategy every quarter with the product champions to update the baseline. This way, the product owners can rearrange the priority of the features if needed.

On top of prioritizing the features for each release, we recommend that the product owners organize each story based on an epic for the project. Each story should include the minimum requirement and has a format such as "As a <persona>, I would like to be able to <feature> so that I can <value>." It's good to include acceptance criteria and features that are out of the scope of that story to avoid any confusion and scope creep. This will resolve the second issue with features being disorganized and lack of requirements for developers.

The third issue can be resolved by establishing stricter change management after the initial baseline has been signed off. Before a change is accepted, the team should conduct an impact analysis and discuss the impact to decide if the changes should be accepted or denied. This would prevent product owners from changing features and feature requirements without team knowledge.

Planning Phase Analysis and Recommendation:

In terms of planning, one key problem area that Collabo faces is the unsuccessful implementation of the Scrum Agile framework across the organization. Further, the lack of understanding and enforcement of standard Scrum Agile methodologies has contributed to problems stemming from the lack of empiricism such as the lack of transparency, inspection, and adaptation. This manifests itself in the lack of historical metrics to measure the team's sprint velocity and the difficulty with estimating the team's capacity when performing sprint planning activities. Also, there is a lack of project management software to capture metrics such as Sprint Velocity within the organization which makes it difficult to capture historical metrics.

Currently, it appears that Collabo is not executing the Scrum Agile framework as per the recommendations in the 2020 Scrum Guide. As per the 2020 Scrum Guide, the scrum team has three roles which includes the developers, product owner, and scrum master. While Collabo has a development team, there is no scrum master or product owner role assigned to the scrum team. According to the scrum guide, "The Scrum Master can serve the organization in several ways including:

- Leading, training, and coaching the organization in its Scrum adoption;
- Planning and advising Scrum implementations within the organization;
- Helping employees and stakeholders understand and enact an empirical approach for complex work."

By having a scrum master who understands the importance of the empirical scrum pillars of transparency, inspection, and adaption, they understand the importance of gathering and benchmark meaningful data such as sprint velocity to support the three pillars. We recommend using a software such as Jira which has built-in Velocity charts which you can utilize to track the Velocity of a scrum team over time.

Another key area problem related to planning is the lack of communication between the product owners and the development team and the lack of change management for introducing new product features. While Collabo maintains a product backlog, product owners are often changing features and feature requirements without discussing with other stakeholders and with little to no documentation regarding the changes. As a result, this makes planning activities very difficult since when the product scope changes frequently and if there is no communication of said changes downstream to the development team, this makes estimating the effort of future releases incredibly difficult. Because of the lack of visibility and the lack of communication of product features, this presents the opportunity of individuals being able to get away with engaging with backroom deals to prioritize certain features over other features.

To assist with the communication and prioritization of product features, we recommend that a product owner be assigned to the scrum team and perform their responsibilities as per the scrum guide. The scrum master can help coach the Product Owner as well as explain their duties to the scrum team. As per the scrum guide, the product owner's responsibilities include:

- "Developing and explicitly communicating the Product Goal";
- Creating and clearly communicating Product Backlog items;
- Ordering Product Backlog items; and,
- Ensuring that the Product Backlog is transparent, visible and understood."

Having the product owner involved in the scrum team and relevant scrum ceremonies is imperative to ensure that the increment of value resulting from each sprint is maximized. By having the Product Owner assist with Product Backlog ordering, this helps the development team prioritize and work based on the most important features first. Also, during sprint planning, if the user stories are missing information or are unclear and are making it difficult for developers to estimate the story points required, the developers can ask the product owner for clarification and resolve any ambiguity that exists. Periodic backlog grooming sessions between the product owner, scrum master, and development leads can be a good way to document, prioritize, and capture any new features into the product backlog. The product owner can then prioritize the new backlog items against the current backlog items based on maximizing the overall product value.

For managing the scope, Collabo should implement a change management system where new major features are discussed amongst a group of decision makers who manage the overall product vision. In other words, this group of decision makers are responsible for assessing new major product features and determining if they should be accepted as part of the product vision, or if they should be left out. Enacting this type of change control can help ensure that the product features that are being developed serve to meet the product vision.

Development Phase Analysis and Recommendation:

Current State Analysis:

Collabo's current development process, while matured, requires an update. As a group, we identified three areas where the process could be improved. The first area will concentrate on

managing code readability and quality. Having a "single unified" coding style not only simplifies development but also serves as a guide for new developers who join your team. The second area will focus on the code review process. Currently, there is no formal structure for code reviews. Once a pull request is pushed, the project supervisor has to accept it before it reaches the production server. Suppose a teammate has a suggestion, they can leave a comment. This process lacks a formal structure, and the inconsistent nature of the reviews suggests that it is insignificant. Finally, the workflow will be a significant area of focus too. After shadowing and reviewing your current processes, we have several suggestions to help you improve your current workflow. For example, things such as upgrades to the current ticketing system and more can be found in the Recommendation section.

Key Area of Focus

1. Managing Code Quality
2. Improvements in Code Review Process
3. Enhancements to Workflow

Recommendations

The first priority will be to manage code quality and in return code readability. We recommend that Collabo's team examine the existing code infrastructure and create a list of the most frequently used coding/design patterns. This list will be used to create a code reference guide for all developers to look back to. In addition, this guide will serve as vital onboarding material for newer developers.

We suggest Collabo create a small workshop to assist in this process. A workshop is an excellent opportunity for senior developers to share their experiences on best practices and give guidelines to show junior developers how to follow proven coding patterns. Also, to effectively address everyone's concerns and ensure a smooth process, clear roles and responsibilities must be given to each member.

Roles and Responsibilities in the Workshop

- Facilitator (1)
- Scribe: (2)
- Senior Developers (3-5)
- Project Managers (1)

The second priority will be to develop a code review process. Collabo conducts code reviews in an asynchronous manner. Developers would submit pull requests and then wait for other developers to provide feedback. This approach, while not flawed, could be improved upon. Before submitting the pull request, the developer must conduct a peer review with another developer. This person is responsible for walking the reviewer through how he completed the project. In return, the reviewer will examine the code to ensure that it meets coding standards and is simple to understand (meaningful and intuitive variable names). The reviewer will also look for improvements and suggest tips if they have any. When these two people come to an understanding, then a pull request is submitted for everyone to review. This process should be light and easy and should typically last 20-40 minutes on average. After the pull request is

submitted, it must be approved by one other developer on the team. In addition, we recommend Collabo develop some automated tasks to handle testing and inspection of code quality. For example, Sonar Cloud is a cloud-based tool that can easily check for simple code quality issues and detect code smells.

The last area of focus will be on improving upon current workflow processes. After interviewing key stakeholders and shadowing Collabo's work process we found several issues in the current ticketing system. Collabo is still using its custom in-house ticketing system for project requests. This was an effective cost-saving solution when Collabo was in the early stage. The system was tailored for a small team of 5 developers. Now, Collabo has over 20 developers and is still growing rapidly. In addition, since the current ticketing system was developed in-house, unexpected errors cause a major bottleneck for the team. Developers can spend weeks resolving errors, delaying needed projects.

We advise Collabo to abandon its internal ticketing system and switch to an online alternative. There are numerous tools available, but we believe Jira will be an excellent solution for all your requirements. Jira not only provides an advanced ticketing solution, but it is also a fully developed cloud-based solution for project management. It is an industry leader in providing easy-to-use interfaces for all facets of management.

Testing Phase Analysis and Recommendation:

Current State Analysis:

One of the main issues that Collabo is facing is that Quality Engineering teams came late in the software development cycle. This caused the testing activities to start late and end late, which often results in projects or releases being delayed. Consequently, QEs were wrongfully blamed for causing the delay in releases.

Another issue with Collabo is that Quality Engineers are often responsible for both designing and building an automation framework for the organization. For example, Collabo needs automation on test runs and release builds to reduce errors and save time. However, this automation effort was not successfully formed because of the resource constraints and the lack of sponsorship from upper management.

Recommendations:

Our recommendation for the first issue is that Collabo should start the testing phase earlier in the software development life cycle to avoid late bugs found. For example, the whole organization should follow the Scrum Agile framework, which includes the testing phase before codes merge to the baseline. This will give Software Testers enough time to find critical issues and allow the developers to fix them before building more code on top of those bugs. By starting the testing earlier, there is a better chance that the projects will be released on time as planned. In addition, it should remove the unhealthy culture of wrongfully blaming the QEs for the delayed in releases.

Furthermore, we would recommend that testers apply the Pareto principle for testing. The idea is that we want to identify which important areas to focus on first so that we can be more

efficient with our time and effort. The Pareto principle can be applied to software testing by directing software testing and automation testing to spend more time working on test cases that really matter to the most valuable customers first. So, software testers should start and focus more on use cases that reflect the needs of the most valuable customers. Edge cases should be considered last during testing a feature. Also, it's very useful to identify the defect clustering since bugs are often clustered in some modules due it their complexity or trickiness. Focusing on those areas first can reduce the number of issues. This should reduce the amount of work for testers and lower the rate of defect reports. Therefore, by starting testing earlier in the SDLC and applying the Pareto principle (aka 80/20 rules), the number of defects should not increase at a faster rate than the number of defects resolved.

The recommendation for resolving the second issue is to have QEs focus mainly on building an automation framework. It's better to have another group to be responsible for the design part to reduce the amount of work for QEs. However, this requires the upper management to support the automation team for what they need to successfully automate those important features such as the test runs and release builds. We strongly recommend that upper managers consider this issue and support the automation team. The reason is because having more automation test suites will make it easier for the teams, and thus the employees will not have to work 60 hours per week. Automation can help catch most of the critical bugs and remove the potential for human error and ensure high-quality software. This will lead to an increase in customers satisfaction with the product due to fewer errors in each release. In addition, automation can ease the issue of having to manually generate a release build.

We suggest that the upper managers consider switching to DevOps if possible since it is an efficient way to boost automation to the next level. Automation is a crucial part in DevOps, and it can make the development process complete a lot quicker and reduce the amount of work for people. A DevOps engineer will also be responsible for introducing automation where possible, testing, and analyzing codes. This can reduce some of the workload from the QEs while making the development process more efficient and more collaborative among the developers and QE team.

Integration and Release Phase Analysis and Recommendation:

Current State Analysis:

One of the main issues with the process is that it is currently manual. When a new feature is released, and ready for deployment, the leads for each development team must manually bring down the current build, upload the new one, install and start the service, and finally perform a smoke test to ensure everything is functioning properly. Having to go through these steps manually every time a new feature is released poses risks for errors due to human interactions when running commands in the system. Manual processes could also result in inconsistencies because developers are relying on their memories to run command lines.

Another main issue with the process is the usage continuity in between releases. For every deployment of a new release, it was noted that the server needs to be down for the duration

of time when the process mentioned above is manually performed. Downtime during deployments means that the business is losing potential customers within that time frame when the software is not able to be accessed by users.

Furthermore, even with downtime minimized, having the system's unavailability is detrimental to businesses because the idle time in which customers are not able to access the system is time loss in productivity and, in turn, potential loss in revenue.

Recommendations:

Below are some of the recommendations our group came up with to address the key issues Collabo is currently experiencing.

- To address the issues with manual processing, we recommend automating the DevOps process. Setting it up as Infrastructure as Code (IaC) and documenting the configurations into a version control tool should achieve three things.
    1. Minimize or eliminate manual process

        This will decrease the time it takes to bring down, upload, install, start service, and smoke test new releases. This will also minimize errors due to human interactions.

    2. Keep history of changes

        This will facilitate reverting back to previous versions if needed during the regression testing

    3. Standardization

        The set of commands are standardized making builds more consistent from each other allowing redundant server builds.

- To address the issue of usage continuity, we recommend building redundant servers. With reference to the previous bullet, having the infrastructure automated and standardized facilitates the ease of spawning of new server builds. This will help with maintenance in between releases and eliminate downtimes because users are still able to access the copy of the same server while the others are maintained.
- The final recommendation our team came up for in integration and release phase is to provide more frequent small releases so that customers can provide continuous feedback in a shorter amount of time.

Defect Tracking Phase Analysis and Recommendation:

Current State Analysis:

The current state of Collabo's defect tracking is an underdeveloped system that is not equipped to handle the current magnitude of reported issues. Due to the system still being in its early stages, they do not have a good way to classify and organize the reported issues. This leads to many repeated issues, poorly classified issues, and untraceable issues. All of this combined is causing the backlog to grow uncontrollably.

Currently, the only way to log problems is through customer support and through Collabo's website users. There also does not appear to be any guidance given to customers when reporting the issues. This results in non-software developers trying to characterize defects in software systems, with no proper guidelines.

Based on the described current state of Collabo's defect tracking the main issues appear to be:

- Poor defect classification
- Inefficient defect log database organization and maintenance
- Inadequate defect logging guidelines and templates

Recommendations:

Based on the above analysis there is a clear path forward, that with minimal upfront investment, can drastically improve the efficiency and success of the current defect tracking process.

Step 1:

The first step involves improving how defects are logged. By providing better guidelines and templates for defect logging we can greatly improve the quality of each report. The best way to do this would be to include guiding questions that must always be filled out when submitting a ticket. Key questions would be: "What feature was being used when the error occurred?" and "What kind of error occurred?". By asking these questions and providing drop down menus with categories to choose from, each defect will have defining features that will be used to classify and resolve the issue. Additionally, the new ticket templates would also require support documents that include software logs of the error occurring. Not only will the errors now be categorized but they will also include all the necessary information for the technicians to begin solving the problem. Providing this solution will have a cascading effect that will aid in resolving the other main issues with the defect logging system.

Step 2:

The second step involves restructuring how the issues are organized and maintained. From step one, we now have a way to categorize each issue which will allow us to organize all of the open tickets by the category they belong to. We can now set up our database to organize each ticket by the feature and the error type. Through having these categories it becomes easier to maintain the system. It is now possible to track all of the errors surrounding a certain feature as well as discard all of the duplicate tickets. This will help to quickly reduce the backlog because all of the repeated issues will be cleared and the man hours required to use the system will be greatly reduced.

Step 3:

Lastly, now that we have a way to categorize, organize, and maintain our defects, we can fully classify them. The classification process will assign priority, possible solutions, and scope to the issue. Since we have organized our defects by the features they involve we can now more accurately estimate the magnitude and scope of this problem. This allows us to assess these parameters, assign priority to each ticket and provide possible solutions.

Conclusion:

Following the provided steps will transform the entire defect tracking process. It will improve it from being a disorganized, mischaracterized, backlogged system to one that is properly maintained, prioritized, and manageable. Investing in these improvement steps will take some upfront cost but will ultimately save money through improving the efficiency and reducing the man hours required to use the system. As a result, we recommend Collabo take our suggestions and integrate it into their work process as soon as possible.

References

Choice, Editor's. “A DevOps Case Study at One of the World's Largest Banks.” *Information Age*, 17 May 2019, https://www.information-age.com/devops-case-study-banks-13771/.

Humphrey, Watts S., Managing the Software Process, The SEI Series in Software Engineering, Addison-Wesley, 1989. (29th Printing, May 2003) (ISBN 0- 201-18095-2)

Santos, Jose Maria Delos. “Pros and Cons of Jira Software for Project Management (2022).” *Project*, 24 Oct. 2022, https://project-management.com/the-pros-and-cons-of-using-jira-software/.

TestProject. “Achieve More with Less: Pareto's Principle in Software Testing.” *TestProject*, 3 Mar. 2021, https://blog.testproject.io/2021/03/03/achieve-more-with-less-paretos-principle-in-software-testing/.

“What Is Devops?” *GitLab*, https://about.gitlab.com/topics/devops/.