Assignment 2 - Shinty Software

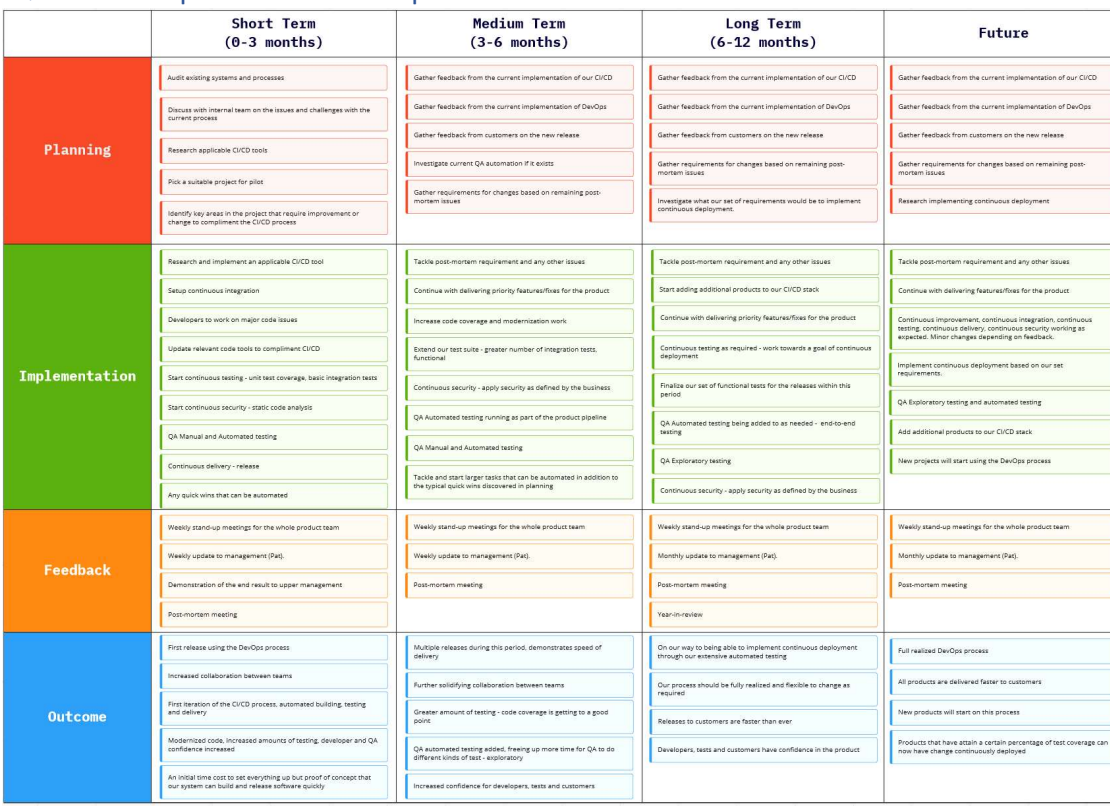
Overview

L00177579

GitHub Repo: https://github.com/L00177579/Assignment2

Might need to zoom in for the images below. Additionally the images are in the GitHub repository.

Q1 - DevOps Process Implementation



First part of this visual design required me to understand who my audience was. Given that this process will be implemented throughout the whole company the audience became apparent to me that it can be anyone within the company. If the document is too high level it wouldn't be of any use to those on the technical side (as they could want to know what is involved specific to themselves in the process) or if too low level upper management and those not in technical roles may not understand part of the process. I think the above strikes an okay balance between low and high levels. It doesn't go into technical details about how certain things will be implemented but it also shows the steps and timeframes.

With this diagram I wanted to make it apparent that in the DevOps process, planning and feedback (or collaboration) are important parts of the process. Each term has its own set of planning steps and feedback events. The feedback from the

previous event should feed into the planning of the next to follow the DevOps idea of continuous improvement. Our systems and processes should be continually worked on as we learn what does and doesn't work for the company.

Shinty Software is a company on decline and they need to get something out quickly. So I've split up the implementation into 3 time periods. For the initial time period of 3 months (short term) we want to have our basic systems in place. Continuous integration and continuous delivery being the important two as we need to get a releasable product delivered to customers. Additionally for future work we need to make a start on continuous testing and continuous security in this time period. At the end of the short term period I'd expect one or more releases. A good amount of manual testing by QA will be required at this stage. In our medium term and long term we're focusing on our testing suites. We need to get our code coverage up, our integration tests covering various areas of our setup, and a start on implementing QA's automated functional testing into the CI/CD process. All of this is towards our goal of applying the left-shift or continuous testing principle to our products and the final goal of continuous deployment.

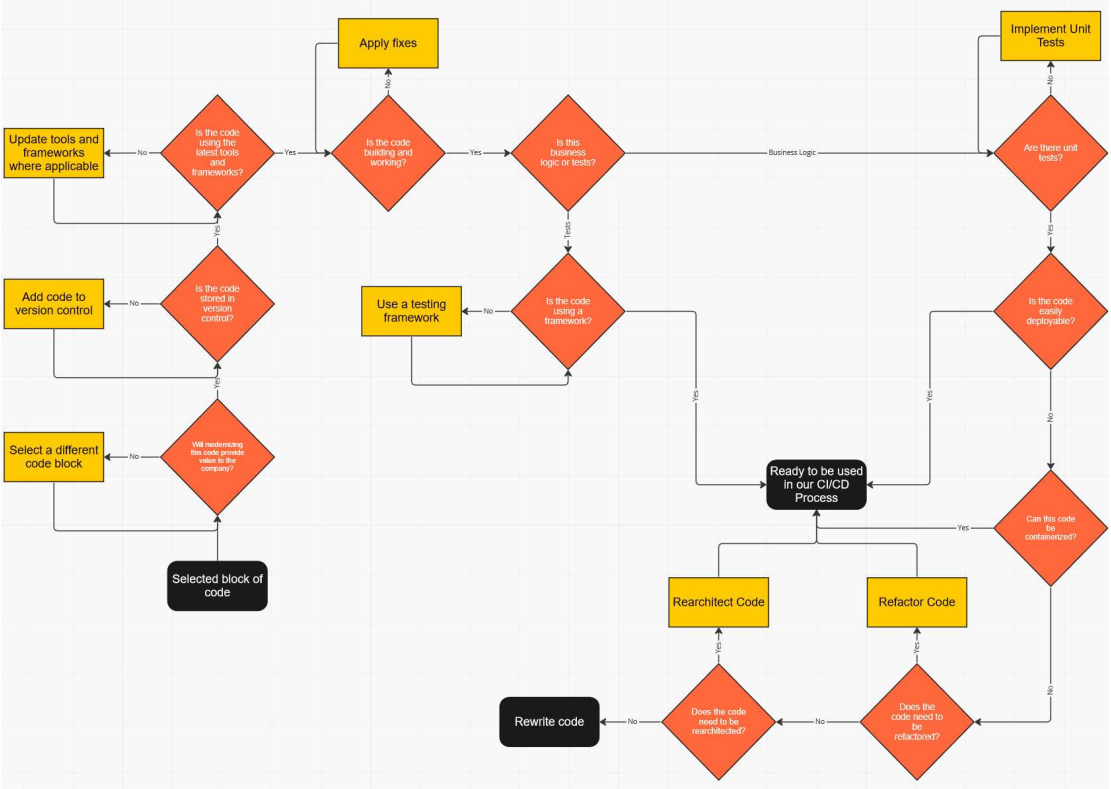
With the lack of staff, we are looking for any quick wins possible during these terms. With the two staff left over to work on what is within this process I would get our junior developer Jalen to work on the unit and integration tests so that they can get more familiar with the code base while our more experienced developer can work on continuing any required code base fixes. This split would eventually drift more to even as we move along the timeline.

Shinty Software has been developing and providing software for 20 years, they know how long it takes to get software out to customers, what issues they continually run into, and the amount of management required for legacy software. So one of the major points in the above diagram is the outcomes section. I wanted to make it clear in the diagram in the outcomes section that we're looking at the end result of all these changes. Upper management or C-Suite executives will want to know why we're making these changes in terms of costs and sales. In this case I've added one of the outcomes being a faster release time. This will improve customer satisfaction and allow the company to charge customers for frequent updates. For developers continuous integration, continuous delivery, and code modernization makes it easier for them to focus solely on the code changes required for features and improves their confidence that changes they make will not break anything else. This is the same for testers, they will have greater confidence that changes coming from development will not break the software. The outcomes of this implemented process are important as they feed into the priority or goal of each type of audience.

I wanted this diagram to be extensible or editable as we went along. I envision this

diagram to be referred to as a simple step by step/process or as a high-level roadmap of what will be achieved.

Q2 - Code Modernization Process



I've gone with a flow diagram for the code modernization process based on my assumption that the audience in question are going to be developers.

As shown in the DevOps diagram - we need to choose a suitable product (and code). There is no reason to modernize code that doesn't require it or isn't going to bring an immediate benefit to the company. The next step is an important one for CI and keeping track of changes - version control. With legacy systems the code could be stored in multiple different locations, we need to centralize those locations around a relevant version control system. Our legacy software may be using outdated tools that are not suitable to modern day processes or are simply security risks. So updating these as necessary is a key step.

For the next part in the flow diagram *Is this code building and working?*, I wanted to cover the case of any code that is a compiled or interpreted language. Either way if there are immediate issues preventing the code from getting past this stage they need to be fixed.

Finally our first major fork in the road. With legacy code there may be some or no testing in the project - if there is we want to be able to handle it in our modernization process. If the code going through the process is a part of testing we need to update the code to use a framework (it's possible one might have been

implemented in-house). Modern frameworks often come with many benefits such as a pretty print of the test results.

If our code is business logic then we need to handle it in a different way. First I want to check if there are unit tests. If there are not, some should be added to the project. The next few decisions revolves around the state the code is in. Can it be deployed? If yes then we should get it through the CI/CD process straight away. If not we have a few decisions to make the first of which is: Can the code be containerized? Containerization allows us to make minimal code changes but still deploy out legacy software to cloud hosting services if we wish (or even locally!). Finally the last two points on refactoring and re-architecturing. For both of these I looked at the topic of technical debt. According to Schmitt (2022) technical debt cost comes in various forms such as: Features taking longer to develop, lower product quality, and loss of company reputation. Given the previous mention in the DevOps process of customer confidence increases we needed to minimize technical debt either through refactoring, re-architecturing or if absolutely necessary rewriting the code.

Q3 - Review Current Code

To begin I ran the code through a pretty formatter and manually corrected the comments that had shifted onto new lines. After reviewing the code I stepped through my flow diagram for modernizing the code.

Will modernizing this code provide value to the company?
For this assignment my assumption to this question is yes.

Is the code stored in version control?
Added to version control to allow for CI/CD.

Is the code using the latest tools and frameworks?
This is a powershell script but it has been noted the latest version of powershell is 7.3.0.0 (*Installing PowerShell on Windows,* 2022).

Is the code building or working?
Powershell scripts aren't built. The current iteration of the script does not run correctly and requires fixes to several areas. Areas marked with TODO require some level of fixes.

Is this business logic or tests?
The file name and from reviewing the code this looks to be a script that is testing the network connectivity between several machines. This could easily be a set of integration tests or system tests. My assumption given the use of a powershell script that this is a system test to check if the network has been correctly configured based on a set of inputs.

Is the code using a framework?
Currently the code outputs an exception upon discovering an incoming port is not open or not added as a trusted host. In its current format it doesn't cleanly print out the list of errors and object information. With large sets of input data this could become unwieldy. For this scenario I recommend using a unit testing framework to write the tests. Powershell has the Pester unit testing framework. With it we can implement ideas such as data-driven tests (this is currently done in the script but is verbose in its implementation of reading in the file and iterating through each object). Additional benefits include the ability to add run-time descriptions to tests and pretty-printed output.

With the code using a test framework it can now be added to our CI/CD process.

Conclusion

The biggest takeaway from designing these processes has been how best to visually display them based on the audience. The audience being more or less technically inclined has a huge impact and what level of detail is required. In the case of my DevOps process I struggled to create something a wider audience can consume. Feedback sessions may change the direction the process is implemented. In contrast I found Code Modernization much easier to deal with due to my development background on legacy codebases.

From the DevOps process diagram it became quickly apparent to me that it is difficult to plan ahead several months in advance. I'd expect that diagram to change drastically as Shinty Software moves through each period. This is to be expected in a continuous improvement scenario. With the lack of staff it is also quite the task to come up with a process that inherently requires many steps to be implemented in very little time.

An additional consideration I made throughout this assignment is on the number of staff that will be involved implementing all these processes. Having an inexperienced developer could also be a time sink for an experienced developers time further eating into the total time available for the implementation step.

On the code modernization side, going through the steps for a powershell script was interesting. It was unclear whether or not the script was used in the code base for a product or was simply a set of system tests. With my modernization process I can easily go through each decision while reviewing scripts to decide what needs to be done to the code to get it ready for the CI/CD process.

Shinty Software is clearly going through some difficult times. Having only three people on a team to implement all of these processes while still being expected to

output a marketable product will be a monumental and stressful task. If I was at this company I would do my best as above to give a process to implement DevOps and Code Modernization but I also wouldn't be too far from dusting the CV off.

References & Bibliography

Installing PowerShell on Windows - Microsoft (2022) Available at: https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-windows?view=powershell-7.3
(Accessed 27 November 2022).

Miller, J (2022) *My Thoughts on Code "Modernization"*. Available at: https://jeremydmiller.com/2022/01/19/my-thoughts-on-code-modernization/
(Accessed 27 November 2022).

Schmitt, J (2022) *Technical debt: how to measure and manage it with DevOps* Available at: https://circleci.com/blog/manage-and-measure-technical-debt/
(Accessed 27 November 2022).