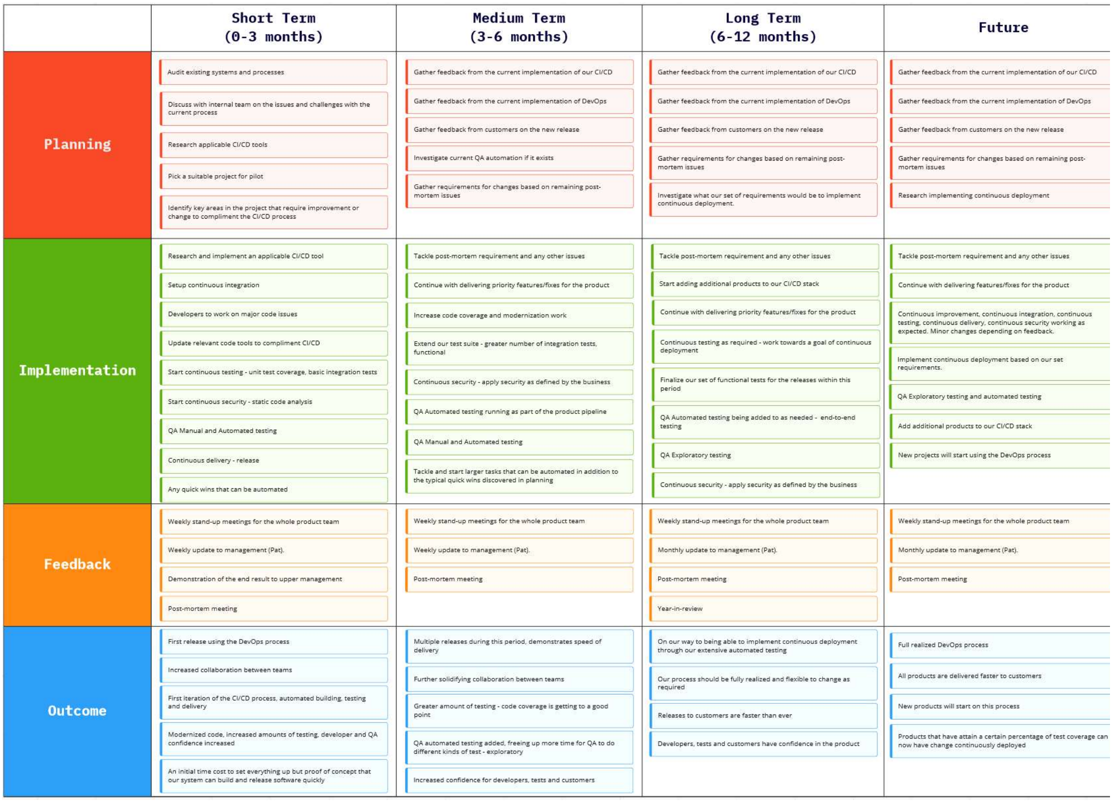
Assignment 2 - Shinty Software

Q1 - DevOps Process Implementation



	Short Term (0-3 months)	Medium Term (3-6 months)	Long Term (6-12 months)	Future
Planning	Audit existing systems and processes Discuss with internal team on the issues and challenges with the current process Research applicable CI/CD tools Pick a suitable project for pilot Identify key areas in the project that require improvement or change to compliment the CI/CD process	Gather feedback from the current implementation of our CI/CD Gather feedback from the current implementation of DevOps Gather feedback from customers on the new release Investigate current QA automation if it exists Gather requirements for changes based on remaining post-mortem issues	Gather feedback from the current implementation of our CI/CD Gather feedback from the current implementation of DevOps Gather feedback from customers on the new release Gather requirements for changes based on remaining post-mortem issues Investigate what our set of requirements would be to implement continuous deployment.	Gather feedback from the current implementation of our CI/CD Gather feedback from the current implementation of DevOps Gather feedback from customers on the new release Gather requirements for changes based on remaining post-mortem issues Research implementing continuous deployment
Implementation	Research and implement an applicable CI/CD tool Setup continuous integration Developers to work on major code issues Update relevant code tools to compliment CI/CD Start continuous testing - unit test coverage, basic integration tests Start continuous security - static code analysis QA Manual and Automated testing Continuous delivery - release Any quick wins that can be automated	Tackle post-mortem requirement and any other issues Continue with delivering priority features/fixes for the product Increase code coverage and modernization work Extend our test suite - greater number of integration tests, functional Continuous security - apply security as defined by the business QA Automated testing running as part of the product pipeline QA Manual and Automated testing Tackle and start larger tasks that can be automated in addition to the typical quick wins discovered in planning	Tackle post-mortem requirement and any other issues Start adding additional products to our CI/CD stack Continue with delivering priority features/fixes for the product Continuous testing as required - work towards a goal of continuous deployment Finalize our set of functional tests for the releases within this period QA Automated testing being added to as needed - end-to-end testing QA Exploratory testing Continuous security - apply security as defined by the business	Tackle post-mortem requirement and any other issues Continue with delivering features/fixes for the product Continuous improvement, continuous integration, continuous testing, continuous delivery, continuous security working as expected. Minor changes depending on feedback. Implement continuous deployment based on our set requirements. QA Exploratory testing and automated testing Add additional products to our CI/CD stack New projects will start using the DevOps process
Feedback	Weekly stand-up meetings for the whole product team Weekly update to management (Pat). Demonstration of the end result to upper management Post-mortem meeting	Weekly stand-up meetings for the whole product team Weekly update to management (Pat). Post-mortem meeting	Weekly stand-up meetings for the whole product team Monthly update to management (Pat). Post-mortem meeting Year-in-review	Weekly stand-up meetings for the whole product team Monthly update to management (Pat). Post-mortem meeting
Outcome	First release using the DevOps process Increased collaboration between teams First iteration of the CI/CD process, automated building, testing and delivery Modernized code, increased amounts of testing, developer and QA confidence increased An initial time cost to set everything up but proof of concept that our system can build and release software quickly	Multiple releases during this period, demonstrates speed of delivery Further solidifying collaboration between teams Greater amount of testing - code coverage is getting to a good point QA automated testing added, freeing up more time for QA to do different kinds of test - exploratory Increased confidence for developers, tests and customers	On our way to being able to implement continuous deployment through our extensive automated testing Our process should be fully realized and flexible to change as required Releases to customers are faster than ever Developers, tests and customers have confidence in the product	Full realized DevOps process All products are delivered faster to customers New products will start on this process Products that have attain a certain percentage of test coverage can now have change continuously deployed

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.

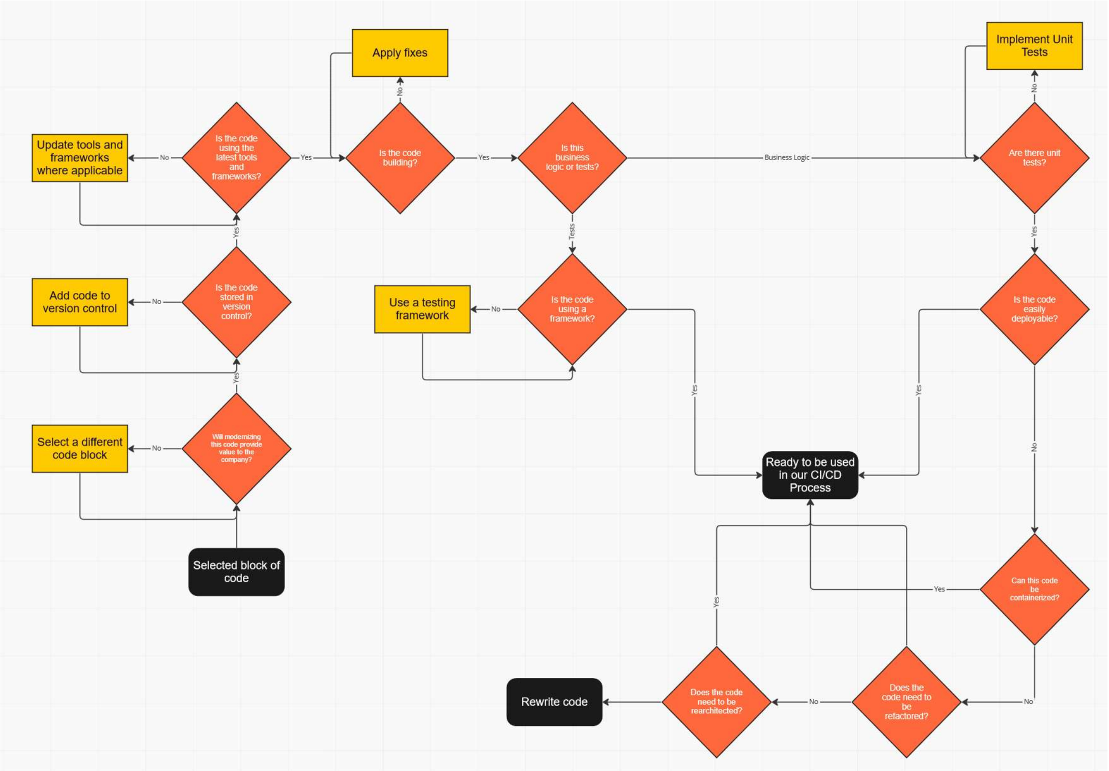
With this diagram I wanted to make it apparent that in the DevOps process, planning and feedback (or collaboration) are important parts of the process. Each term has it's own set of planning steps and feedback events. The feedback from the previous event should feed into the planning of the next to follow the DevOps idea of continuous improvement. Our systems and process 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 the 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. On 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.

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 allows the company 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


Apply fixes
Implement Unit Tests
Update tools and frameworks where applicable
No
Is the code using the latest tools and frameworks?
Yes
No
Is the code building?
Yes
Is this business logic or tests?
Business Logic
No
Are there unit tests?
Yes
Tests
Yes
Add code to version control
No
Is the code stored in version control?
Use a testing framework
No
Is the code using a framework?
Is the code easily deployable?
Yes
Yes
Yes
Select a different code block
No
Will modernizing this code provide value to the company?
No
Ready to be used in our CI/CD Process
Selected block of code
Yes
Yes
Can this code be containerized?
Rewrite code
Does the code need to be rearchitected?
No
Does the code need to be refactored?
No

Tes

Q3 - Review Current Code

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. In addition it's difficult to plan a whole process several months in advance of the supposed end. Feedback sessions may change the direction the process is implemented. If we contrast

References & Bibliography

Appendices