nextern®
Medical Device Enabled

MODERN PRACTICES TURBOCHARGE MEDICAL DEVICE SOFTWARE ENGINEERING

www.nextern.com

651.203.2100

Contact Us

SUMMARY

Nextern's software engineering team focuses on developing robust, easy to use solutions for connected medical devices and digital health management. Adapting modern software engineering practices to the rigor of the regulated processes required by the industry and international standards has challenges but enables more responsive and rapid software solution development.

INTRODUCTION/BACKGROUND

Software development for regulated industries requires rigorous documentation and detailed processes to ensure the highest quality software. This has traditionally been achieved through meticulously structured development using Waterfall methods which generate extensive documentation at each step.

Modern software development practices such as the Agile development methodology, test-driven development (TDD), DevSecOps, and mob or pair programming (also known as ensemble programming) can enable more responsive, higher quality and efficient software delivery.

Nextern's software engineering team is implementing processes to overcome the challenges of leveraging these software development practices to improve delivery of regulated medical device and software applications.

PROBLEM

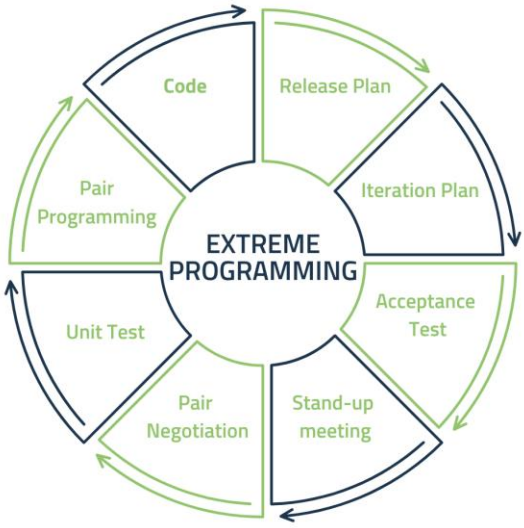
Medical device and medical software application development requires generating many artifacts to satisfy requirements of regulatory bodies and industry standards. This overhead can slow down and increase the expense of software development. Often, software teams don't think they can leverage modern ways of developing software due to the constraints of these regulations. Sometimes, they are correct. Modern methods do not lend themselves to use in a regulated environment without some work to ensure they are used in the right way and at the right time. Adjustments may also be needed to these practices so that they generate the necessary documentation required by regulations.

PROBLEM STATEMENT: The problem lies in the inherent challenges faced by medical device and software development teams. Regulatory requirements and industry standards impose a substantial burden of documentation. This can hinder the adoption of modern software development methods, making it difficult to leverage cost and speed improvements these methods can provide.

SOLUTION

Nextern is adapting modern software engineering practices, specifically Extreme Programming, test-driven development (TDD), mob programming, and DevSecOps for use in the regulated medical device software application domain.

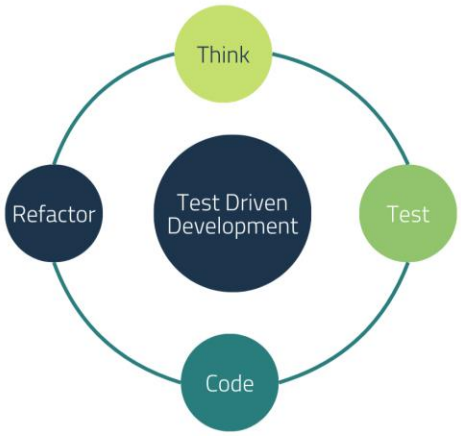
Our implementation of **Extreme Programming** brings our end users to the forefront of our process and allows us to respond quickly to changing project and product requirements. We're changing our process to allow us to create incremental documentation under strong change control. This lets us capture changes as we go and make sure that nothing is lost. Working at a sustainable pace means our team can maintain focus and attention to detail. Our regular retrospectives let us adapt processes and practices as needed.



Our team typically executes using a structure of month-long releases that are planned around the implementation, verification, and documentation of a specific feature. We include key stakeholders - right now product owners, but eventually also clinical, regulatory, design assurance, and other functional groups - in planning.

Each release is broken into a series of week-long iterations where the team, with stakeholder input, plans, implements, verifies, and documents vertical slices of the feature.

Test-driven development (TDD) challenges our software engineers to think about the testability of their code from the beginning. We expect TDD to help us verify safety-critical code and drive us to structure code modularly, making it more robust and easier to maintain. We are implementing DevOps pipelines to ensure that the testing we create is executed and eventually automatically documented throughout the development lifecycle.



TDD is becoming our preferred way to create software. The goal is for developers, often working in a pair or a mob (more on that later), to write the unit level tests for a feature or function before they begin writing any code. Software engineers submit the tests and code together for review before the changes are integrated into the code base. Once integrated, the tests run on each build we execute.

Mob programming, or ensemble programming, promotes knowledge sharing within our software team and across teams such as design assurance. Mob programming sessions cut out the back and forth of a typical pull request review process. Working together, our teams own the entire codebase and documentation. Safety-critical functions and quality are a team responsibility. Having the eyes of multiple engineers and, when possible, stakeholders on the tests and code as we write them is making us better at catching all test cases, writing easier to understand code, and capturing essential documentation.

We are developing ways to take credit for the mob review sessions while minimizing overhead. That will allow us to provide evidence of the reviews performed on the software as is often required by regulatory authorities.

DevSecOps will enable efficient implementation of many of our practices. Automating the generation of incremental documentation will help us capture changes to requirements and flag tests that need to be updated or re-executed. Using CI pipelines for our builds allows us to automate the integration of code changes and shift our security and quality practices left

by automatically executing static analysis and running tests from our TDD practices on each build.

In the future, we will also configure these pipelines for release builds and include additional functions such as generation of version description documents, test results reports, packaging of source code and details of the changes integrated into a release. For mobile applications, we hope to even include the release of artifacts to the relevant app stores.

We intend to create similar automation to support mob programming, capturing the details of each session and collating them into final reports to record the team that wrote, reviewed, tested, and approved the code changes.

OUTCOME

At **Nextern**, software engineering for medical devices and software applications is a team sport. We are committed to using the best practices of modern software engineering to solve problems for our customers and ultimately for patients. Regulatory requirements are not a barrier to using these practices. With a little adjustment and occasionally some additional record-keeping, we have started to leverage these practices to increase our efficiency in creating high quality, safe software and all the documentation needed to demonstrate that to regulatory bodies. As importantly, our use of these practices has helped us build and maintain a high quality, dedicated team of software engineers who thrive on tackling difficult problems and are happy working in a regulated environment. Like any team, this is an ongoing journey for us, and we expect to continue refining our practices, investing in new tools, and seeing more improvements.