Tech Mahindra

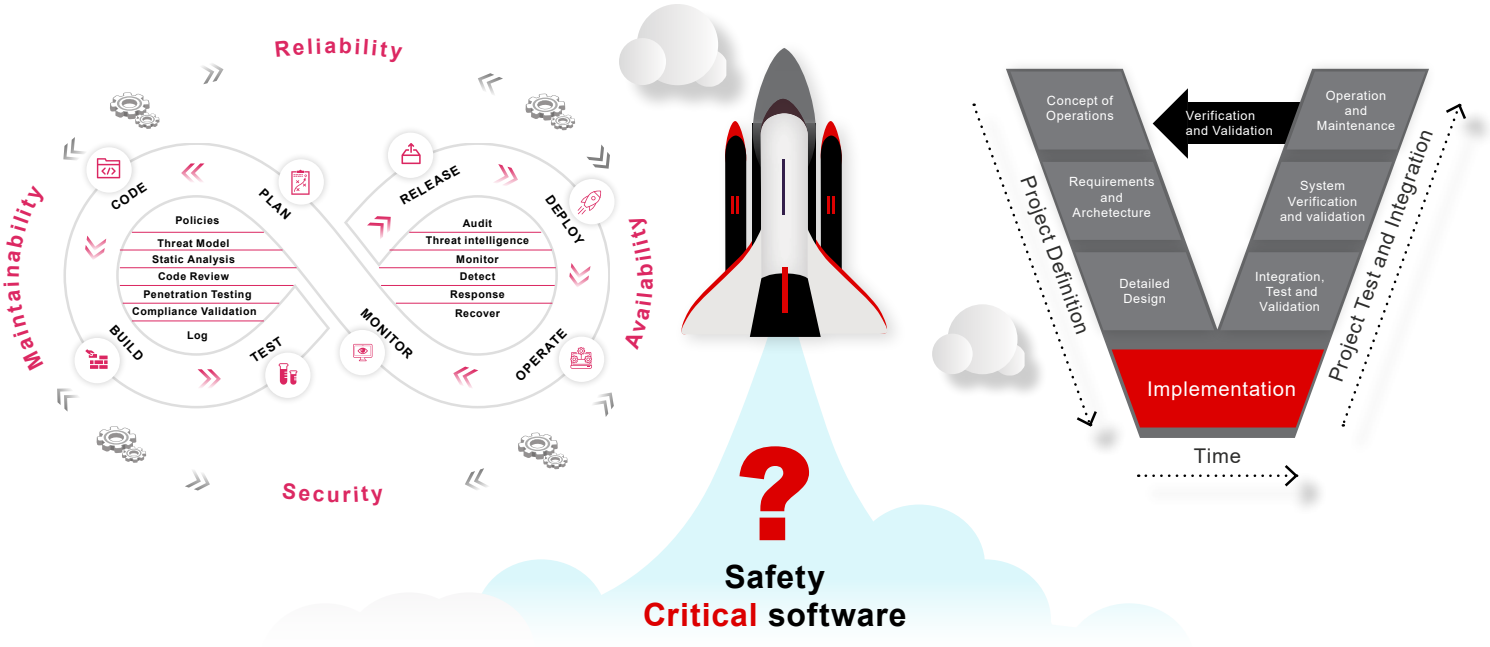
Safety Critical Software Development

Applicability and Adoption of DevSecOps & Continuous Testing

Authored by
Anjali Chhabra Nandwani

Connected World.
Connected Experiences.

Safety Critical Software Development
- Applicability and Adoption of DevSecOps & Continuous Testing



There is a huge shift in IT organizations to adopt DevSecOps practices – while there can be many ways to define DevSecOps; simplistically put, an approach to software delivery where teams that develop and test the system / software, teams that are responsible for ensuring production operations for the developed systems / software and teams that are responsible for ensuring secure operations, all come together and work cohesively to deliver high quality and highly secure software at a high velocity for smooth production operations to business. Increased usage of automation to speed up the lifecycle is another key aspect for DevSecOps adoption. Constant feedback loop from production environment (continuous monitoring) and end users back to the development and QA teams is also crucial for DevSecOps success.

Now let us also understand what a Safety Critical System is – in simple terms, a system whose failure can cause significant loss like severe injury or death, property / equipment loss or harm to the operating environment and surroundings. A Safety Critical System is a combination of Hardware, Software, Connectivity / Operating Environment and human participation.

For the purposes of this document we will keep the conversations around the delivery of Software / Firmware for the Safety Critical Systems. Some of the common examples of such SW are the ones that are part of Aviation Industry (e.g. Air Traffic Control, Engine Control, etc.), Healthcare Industry (e.g. Robotic Surgery, Ventilation systems, MRI machines, etc.), Automotive (e.g. Advanced Driver Assistance Systems, etc.), Rail Road Industry (e.g. Positive Train Control, Track Management, etc.) and so on.

Most of the times the development and testing efforts for a safety critical software follow very traditional V-Model driven Dev-Test methodology. While the industry is rapidly moving towards DevSecOps, Agile development and Continuous Testing & Quality Engineering practices; how easy is it to adopt these practices for Safety Critical Software? Is there any conflict? Can DevSecOps and Continuous Integration / Continuous Testing / Continuous Deployment be directly applied to Safety Critical Software or are there any limitations?

Let's discuss the Conflicts first...

Methodical Planning vs. Change is the only constant

Some crucial requirements from compliance and regulatory standpoint for Safety Critical Software expect the teams to be able to provide evidence at any point in time of:

- **Exactly what has been delivered**
- **How the deliverable was produced**
- **Traceability of Validation & Verifications artifacts to Requirements and Design artifacts**
- **Ability to reproduce the deliverable and show evidence of repeatability and reliability of Validation & Verification results**

Now one aspect of DevSecOps which has some conflict with the above is that agile methodology is typically followed here and many times agile gives liberty to the teams to define and refine delivery processes as they move along. Guidelines are defined but strict adherence to a set of guidelines is not mandated. Many times the compliance and audit teams responsible for ensuring the quality standards and regulatory expectations for safety critical systems, see adoption of agile as a risk for such systems. Most of the times due to this fear, Safety Critical Systems team end up adopting very traditional V-Model type of approach to development and delivery.

Now if we actually try to identify synergies between aspects of DevSecOps, Agile, Continuous Testing, CI/CD and the Safety Critical Software development, there are quite a few...

Modular Design and Development

DevSecOps promotes the culture if modular design and development. The whole idea is to break the monoliths and look at the concept of micro services to deliver Minimal Viable Product (MVP) faster to the market.

Definitely this approach aligns very well with the aspects of Reliability, Availability, Maintainability and Scalability (RAMS) demands of Safety Critical Systems. The ease with which the RAMS aspects can be validated and verified for modular components and services is much better as compared to large monolithic systems.

Also, identification of issues and bottlenecks is much easier with modular designs. Traceability, which is one of the key compliance ask for Safety Critical Systems is also much granular with modular design and development approaches. All in all this aspect of DevSecOps model has clear benefits for ensuring quality of Safety Critical Systems.

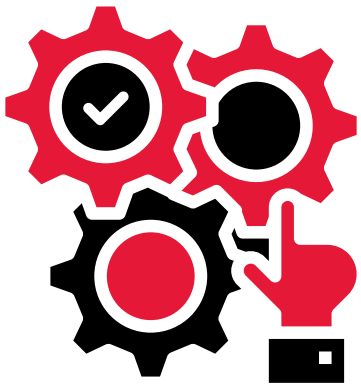
Shift Left & Continuous Integration

The biggest challenge that many Dev and Test teams working on Safety Critical Systems face, is the issue of finding many integrations, system reliability, safety, compliance and security related defects during the Field Testing, Field Integration Testing and Operations Testing phases.

Many a times, these defects originate from the requirements and design phases; and fixing of these issues can lead to a complete change of development and validation approaches, thus causing major loss of dollar and effort.

Concepts of Continuous Integration and validation throughout the development phase by using virtualization of data and services, can definitely help in early detection of many such defects. Concept of Reliability Engineering by incorporating component level reliability aspects in the DevOps cycle and Static Analysis of design and code for early detection of reliability and security issues, are the core benefits that can minimize the impact of such issues during the later phases of SDLC.

Automated Verification & Validation



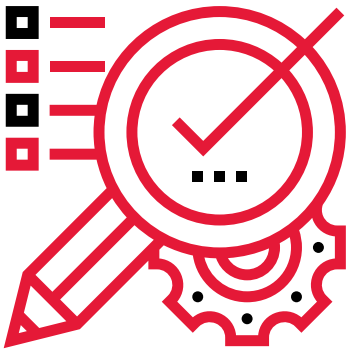
Verification and Validation is the core focus of any software and a critical and utmost important phase for Safety Critical Software. Think about manually verifying all standard flow, exception paths, alternative flows; and hazard and safety specific scenarios for safety critical software and tracing all verification results back to the requirements and design – this can be a humongous task.

Modern development and test practices rely on Model Based Testing and Automated Test Execution integrated with Application Lifecycle Management tools for automated Test Design, automated Test Execution and automated traceability generation. Definitely this can take away a lot of manual efforts from the development and testing cycle; and also traceability management can be automated to a great extent.

Model Based Testing (MBT) is a specific area that has a lot to offer for Safety Critical Software Testing space. Ensuring proper modeling of safety specific features and requirements and then ensuring complete coverage through leverage of intelligence built in MBT tools, can improve the overall quality of the safety critical software. Automated test cases and scripts generation and traceability to models can all add significant value

Automation of Repeatable & Mundane Tasks

Another crucial aspect, that is almost a given expectation from DevSecOps model, is the focus on automation. When it comes to safety critical systems, it is not just the functional and non-functional testing aspects that are important from quality standpoint but areas like code reviews against the standards like IEC 61508 OR EN50128, traceability of every aspect of verification back to the requirements, design and code, evidence of verification results in form of reports that can many times be valid for years, are all very critical.



Automating the areas like standard adherence to code through automated code quality check tools, traceability management through an ALM tool completely integrated with DevTest pipeline, Automated Reporting and Dashboard solutions completely integrated with automated test execution suites, are all aspects that can be leveraged to ease some of the mundane, but crucial asks from compliance and regulatory standpoint for the safety critical systems.

This kind of automation also helps in avoiding some embarrassing situations with auditors caused due to human oversight, at times.

Static Analysis and Early Security Checks

While we touched upon the static analysis aspects in the Shift-Left and Continuous Integration section, this subject requires some detailed discussion as this can play an important role for ensuring the quality of Safety Critical Systems. There are specific code development best practices and compliances that Dev teams are expected to adhere to, for producing secure code.

While there are generic guidelines for secure code. there are also industry specific aspects, for example: DO-326 / ED-202 that has standards defined for ensuring airworthiness and information systems security guidelines and cyber security guidelines for IT & OT aspects of the aviation systems. Now, while manual code reviews, peer reviews, configuration and environment checks are possible, they are time consuming and can be error prone. With the advancement of automation tooling, there are tools that seamlessly integrate with entire delivery pipeline, starting from the IDE that developers are using for coding to monitoring and analysis of production logs.

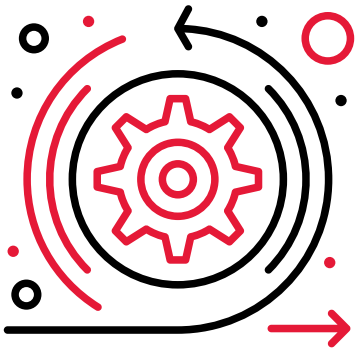
These tools can take over a lot of manual tasks to ensure compliance to set security standards and can enable early detection of vulnerabilities that can cause major security leaks, or can help in log analytics to ensure preventive measures. Definitely, this aspect of DevSecOps enablement is very helpful for all types of software systems but is a greater help for Safety Critical Systems.

How to incorporate best practices of DevSecOps, Agile, CI/CD and Continuous Testing for Safety Critical Systems

Considering there are some conflicts and so many synergies of DevSecOps and Continuous Testing adoption for Safety Critical Systems, it is imperative to define an approach where Safety Critical Systems software can benefit from all the best practices and advanced automation tooling and techniques of DevSecOps model, without compromising on the critical aspects of RAMS as well as regulatory and compliance.

Based on all our experience in working with many of our customers in Aviation, Rail Road, Medical Devices and Healthcare Equipment industries that pioneer Embedded Software development and testing for Safety Critical Systems, the models and strategies that we have seen working in our deliveries, are defined in the next few sections.

Hybrid approach – V-model and Agile / CI-CD

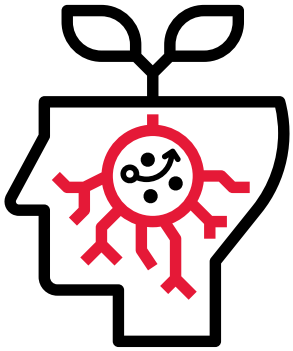


Considering that traceability from phase-to-phase is an important aspect of Safety Critical Software, V-model seems to be an obvious choice as it necessitates the completeness of all dependencies from the previous phase before starting with the next phase and thus leading to complete traceability links without any doubts.

Now this approach is a deterrent for agile adoption, however having an iterative and incremental approach within the development phase and CI/CD and Continuous Testing are the aspects that can be easily adopted without deviating from the basic guidelines of compliance from documentation and evidence of phase completion etc.

Inception, Planning and Requirements phase

For any Safety Critical System software, business specifications and feature / requirement aspects are generally well defined and as a principle cannot be left ambiguous, considering the impact an ambiguous requirement can have on such systems.



It is not possible to embark on a journey of developing a safety critical system software without well-defined requirements and comprehensive plan. Again, it's not compelling that there can be no deviation to the defined and documented requirements; however, from the baseline requirement, any change has to go through an agreed upon change-management process that is well documented and quality engineering processes should have traceability to the original requirement, along with any changes to the same.

Following the V-Model type guidelines for this phase is helpful and critical for the overall quality.

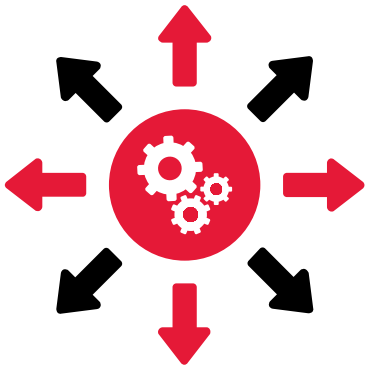
Development and Continuous Testing

Development and testing phase for Safety Critical Software can definitely adopt Agile and CI / CD and following can be some of the key guiding principles for this phase:
Iterative development of features and MVP delivered to testing phase faster

- **Component level tests for early fault detection**
- **Static Analysis for code quality coverage**
- **Early integration tests enabled in development phase through usage of techniques like data and service virtualization**
- **MBT integrated with test automation and ALM tools, for end to end automated test design, test execution and traceability**

Continuous Integration and Continuous Deployment

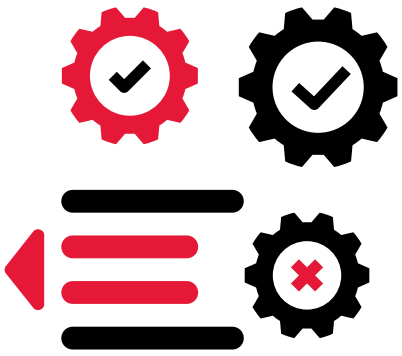
Once of the biggest challenge for traditional V-Model driven SDLC is that integration of all components that build the software, are aligned towards the end of the cycle, leading to multiple issues being detected very late in the lifecycle. For Safety Critical Software as well, traditionally the big bang integration towards the end has been the norm for very long and most of the critical integration issues, performance and security issues as well reliability and scalability issues, all gets detected towards the end of the development cycle.



CI / CD adoption, where smaller subsets of system are developed and released for lab testing is crucial for overcoming the challenges of the traditional "big bang" delivery approach. Ability to simulate parts of the software / system that is not ready but still tests the end-to-end deployment in a lab environment or test bench can also help in identifying not just functional issues but some security and performance issues as well, much early in the lifecycle.

Though most of the times, end-to-end field tests and operational tests still are extensive and time consuming process and cannot be eliminated for Safety Critical Software; CI / CD adoption and early integration of subsets of software for deployment and testing in lab environment can definitely reduce the number of defects that are found in later phases. This can significantly reduce the rework effort and make the entire process much more efficient.

Shift-left of some aspects of Operational Acceptance Testing



Operational Acceptance Testing is a critical phase of safety critical software and covers many non-functional aspects to ensure production readiness of the system / software. Typically, this phase has limited functional testing (mainly limited to the scenarios that are needed to test non-functional aspects). Core focus is on aspects like recovery, supportability, portability, usability, scalability, reliability and security. Again, most of the times, these types of testing is pushed towards the later stages of SDLC and just before the release of software to production; however, with agile adoption, many aspects of this testing can be run in a much more iterative manner, giving all stakeholders more time to validate and adjust expectations.

This is a very vast topic, however examples such as DevOpRET technique mainly focuses on Continuous Software Reliability Testing in DevOps, which is possible due to close collaboration of Dev and Ops teams, thus giving access to development team on the operational data, critical for reliability engineering. Other examples of shift-left of operational acceptance tests include Static Analysis for security vulnerabilities and component level penetration tests, component level load and scalability tests.

Conclusion

To conclude, it is not difficult to adopt best practices and accelerators from DevSecOps, Agile, CI/CD and Contiuous Testing for Safety Critical Software development and testing. The key is to create a comprehensive strategy of leveraging the guiding principle from documentation and compliance standpoint from the V-Model; and then implement best practices from modern development strategies to deliver high quality safety critical system efficiently.

It is important to understand that Safety Critical Software Development and Testing needs the rigor and discipline of V-Model and best practices of agility and iterative development and testing, CI / CD, Continuous Testing, Shift-Left of Non-Functional (performance, security, reliability) testing aspects from DevSecOps principles.

Aspects of change management, documentation and traceability cannot be ignored but need to be automated to make the process efficient.

At Tech Mahindra, our Digital Assurance Practice & Quality Engineering team has been working on various accelerators and frameworks, based on these principles that can help with increased DevSecOps and Continuous Testing adoption for Safety Critical Software development and testing. These accelerators are part of our AI powered Intelligent Automation Platform LitmusT.

Anjali is a technology leader with 20+ years of experience in Information Technology Services industry, with specialization in DevSecOps, Quality Engineering and ADMS Services.
She has worked with large enterprise customers ranging from Banking and Financial Services, Telecommunication, Technology, Retail & Consumer, Healthcare and various industry verticals. She has proven track record on consulting for DevSecOps, Scaled Agile and Quality Engineering Transformation initiatives, Automated Delivery Pipeline (CI/CD/CT) framework design and set-up for large enterprises, Quality Engineering Community of Practice (COP) set-up, Program Management and Governance. In her role, she has worked with diverse IT teams globally to optimize SDLC with usage of Predictive Analytics, Machine Learning and RPA. She comes with in-depth understanding of Integrated SDLC Automation experience along with framework design / tool selection and ROI analysis.

Anjali Chhabra Nandwani
VP & Head Digital Assurance Services
Tech Mahindra Americas
LinkedIn

Tech Mahindra

Tech Mahindra, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for informational purposes and private circulation only and do not constitute an offer to buy or sell any securities mentioned therein. They do not purport to be a complete description of the markets conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Unless specifically noted, TechM is not responsible for the content of these presentations and/or the opinions of the presenters. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.

www.techmahindra.com

connect@techmahindra.com

www.youtube.com/user/techmahindra09

www.facebook.com/TechMahindra

www.twitter.com/Tech_Mahindra

www.linkedin.com/company/tech-mahindra