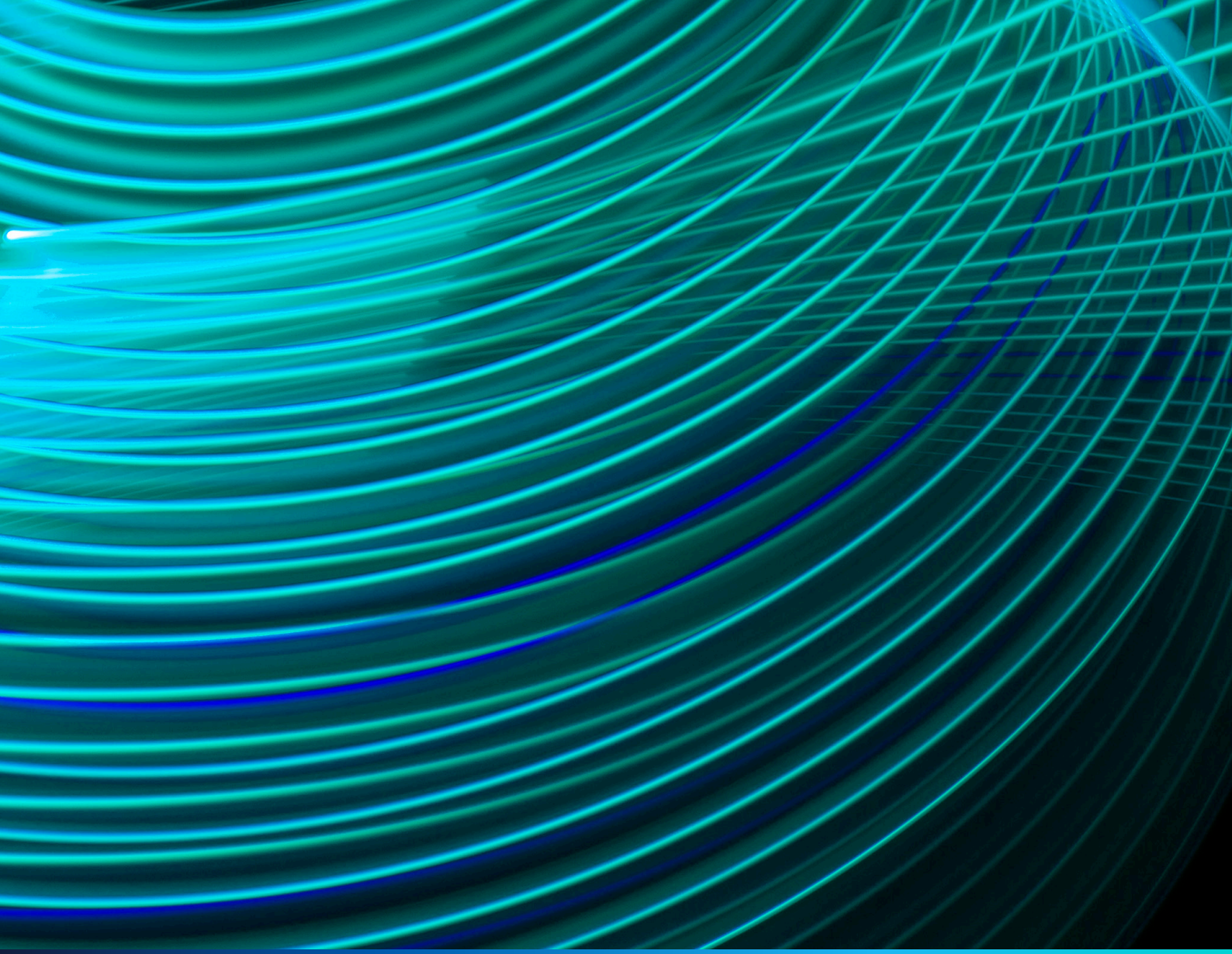
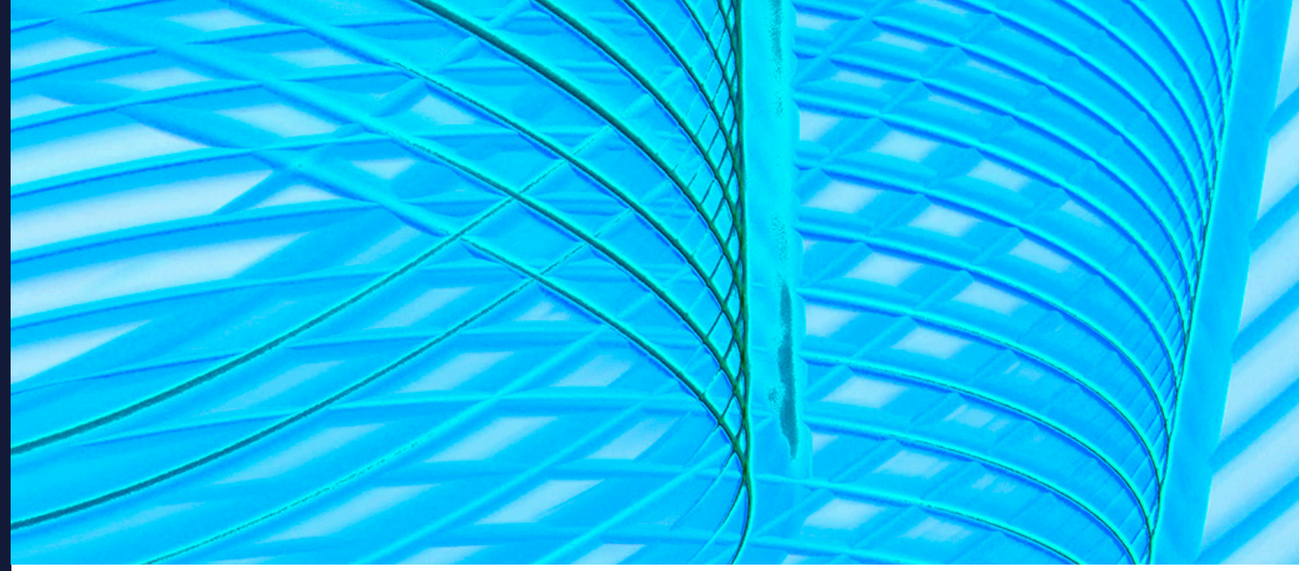


From Automotive *to MedTech:*

Proven Software Development Practices as Blueprints for the Next Wave of Innovation.

Capgemini

01 Contents



02 Executive Summary

The medical technology (MedTech) industry is undergoing a profound transformation as software becomes a central element of product differentiation, clinical performance, and regulatory compliance. At the same time, companies face growing development complexity, increasingly stringent regulatory requirements, and rising expectations for faster innovation cycles. Balancing speed, quality, and compliance across the software lifecycle has therefore become one of the most critical challenges for MedTech companies.

This whitepaper examines how proven practices from the automotive sector can help address these challenges, focusing on two key enablers: the adoption of a Software Factory approach and the responsible integration of Open Source software and Generative AI. The Software Factory model introduces an industrialized, standardized, and highly automated development environment built on reusable components, integrated toolchains, and continuous quality controls. By embedding compliance and traceability directly into development workflows, it enables significant improvements in efficiency, consistency, and regulatory robustness.

In parallel, the paper explores how Open Source strategies and Generative AI can be leveraged to accelerate innovation and productivity while meeting the specific demands of a regulated MedTech environment. Drawing lessons from automotive implementations, it outlines governance models and riskmitigation practices that allow MedTech organizations to benefit from these technologies without compromising safety or compliance.

Combined, these approaches offer tangible benefits, including faster time to market, reduced development and maintenance costs, higher software quality, and stronger alignment with standards such as IEC 62304, Quality Management Systems (ISO 13485), and MDR/FDA requirements. Supported by real-world case studies, this whitepaper provides MedTech executives and technical leaders with practical guidance for modernizing software development and maintaining competitive advantage in an increasingly software-driven industry.

03

Current Challenges in MedTech Software Development

MedTech software development faces a unique set of challenges driven by the industry's highly regulated and safety critical nature. Regulatory complexity significantly slows down development, as software must comply with strict standards such as IEC 62304 for medical device software lifecycle, ISO 13485 for quality management, and regional regulations like FDA guidelines in the U.S. or MDR in Europe. These frameworks require extensive documentation, risk management, verification, and validation, which adds considerable time and cost to every release cycle.

At the same time, many organizations rely on heterogeneous toolchains for example, mixing manual spreadsheets for requirements with separate testing platforms and disconnected version control systems. Combined with legacy practices, this results in limited process standardization and inefficiencies across development teams, making collaboration and scalability difficult.

Ensuring end-to-end traceability, from initial requirements through design, implementation, verification, and release, is mandatory to prove compliance, but often challenging without strong automation. For instance, linking a software requirement to its corresponding test case and risk analysis can involve manual effort across multiple systems, increasing the risk of gaps and audit findings.

Additionally, the adoption of modern technologies such as DevOps pipelines, containerization, or AI driven testing is frequently hindered by risk aversion, long validation cycles, and concerns about regulatory acceptance. Even when these technologies promise faster delivery and improved quality, MedTech organizations often hesitate due to uncertainty about how regulators will interpret their use. This cautious approach makes it harder for the industry to innovate at the pace seen in less regulated sectors like consumer electronics or automotive.

04

Software Factory Approach

4.1 Concept and Definition

The automotive industry is undergoing a profound transformation driven by the rise of Software-defined Vehicles (SDVs). Modern vehicles are no longer defined solely by mechanical components. They rely on complex software ecosystems that enable advanced features such as autonomous driving, connectivity, and electrification. This shift has introduced challenges, never experienced before in software development, making traditional methods inefficient and insufficient.

4.1.1 What is a Software Factory?

A Software Factory is a structured and systematic approach in software development that applies principles of industrial production. It uses automated processes, reusable components, and a consistent development environment to build software efficiently, accurately, and at scale, similar to how a physical factory organizes and streamlines manufacturing.

- **Clear, testable requirements:** Focuses on user-centric, verifiable specifications to streamline validation.
- **Standardized development environment:** Ensures consistency across teams and projects.
- **Automation and scalability:** Integrates CI/CD pipelines, automated testing, and deployment to accelerate delivery.
- **Ensured quality:** Aligning the automated processes with standards like ASPICE, FuSa and SOTIF ensures the compliance of the software with these standards without slowing down the software development process. This includes an end-to-end monitoring of all artifacts, including full traceability to the requirements.
- **Reusable assets and organized processes:** Promotes modularity and reuse of components to reduce effort and improve reliability.

4.1.2 Why are Software Factories needed in Automotive Software Development?

Need for New Methodologies

The transition towards Software-defined Vehicles calls for faster, more structured approaches. Automotive and other high-tech sectors highlight the urgency for methodologies that enable scalability, automation, and quality assurance.

Modern Software Complexity

Automotive software today consists of numerous modules, each containing thousands of lines of code. Development is carried out by large, distributed teams, often across multiple suppliers and regions. This scale resembles industrial production lines rather than small, isolated projects.

Variant and Version Management

Car manufacturers use common software platforms across different models, configurations, and even brands. Managing these variants efficiently and integrating components of different car generations together requires structured processes to ensure consistency, compliance, and timely delivery.

Limitations of Traditional Methods

Legacy development cycles are slow, costly, and lack the agility needed for modern projects. They cannot meet the stringent requirements for speed, precision, and regulatory compliance demanded by today’s automotive applications.

4.2 Software Factory in Automotive

To visualize a Software Factory, it can be envisioned as a modern, automated production system, similar to a physical factory, where all contributing roles (requirements engineers, developers, testers, QA) continuously feed their artifacts into a unified, orchestrated workflow. Instead of manually created, isolated work products that only come together late in the project (e.g., at integration or release milestones), the SWF operates like a conveyor belt: progress depends on continuous input, and automation ensures that integration, verification, and validation happen early and often. This approach delivers several benefits:

1. **Shift-left** – Issues are detected early, reducing rework and improving predictability.
2. **Quality gates** – Automated checks and compatibility validations ensure quality targets are met continuously rather than at the end.
3. **Performance and functional testing** – Continuous evaluation gives early insight into whether the system behaves and performs as expected.
4. **Reporting and transparency** – Management gains deep, real-time visibility into software maturity, risks, and progress.
5. **Speed and scalability** – Cloud-based, incremental builds run in minutes instead of hours, enabling developers to build frequently and understand the current state of the software at all times.

An automotive Software Factory is designed as a highly organized, automated ecosystem that ensures efficiency, quality, and compliance throughout the development lifecycle. At its core, it integrates Continuous Integration and Continuous Deployment (CI/CD) pipelines to enable rapid, iterative builds and deployments across large, distributed teams. Automated compliance checking is embedded into these pipelines to guarantee adherence to industry standards such as ISO 26262 and ASPICE without slowing down development. To manage the complexity of variants and configurations, which is a common challenge in automotive where multiple models and brands share a software platform, the factory employs advanced configuration management techniques that maintain consistency and reduce errors.

Furthermore, virtual integration and validation using digital twins accelerate testing by simulating hardware and system behavior early in the lifecycle, minimizing dependency on physical prototypes. This approach is complemented by comprehensive testing strategies and scenario creation capabilities. An automotive Software Factory can incorporate automated test case generation tools that is able to create thousands of test scenarios based on requirements and use cases. These scenarios cover a wide range of driving conditions, edge cases, and potential failure modes, ensuring thorough validation of the software.

Additionally, the factory leverages data from real-world driving to continuously refine and expand its test scenarios, creating a feedback loop that enhances the overall quality and reliability of the software. A typical ADAS ECU development project for example can easily reach 10.000 unit and component tests which need to be tested and verified. This does not even include HIL testing or scenario-based functional tests of the ADAS algorithm. Such an amount of test cases requires a robust testing and scenario creation framework, which not only improves the detection of potential issues early in the development cycle but also supports the validation of complex autonomous driving features and safety-critical systems.

A robust traceability framework underpins the entire process, ensuring end-to-end visibility from requirements through implementation and testing. This traceability is critical in automotive development, where safety, compliance, and auditability are non-negotiable. Together, these components create a scalable, repeatable, and quality-driven environment that transforms software development into an industrialized process – meeting the demands of modern Software-defined Vehicles.

4.3 How can MedTech adopt a Software Factory Approach

To successfully adopt the Software Factory paradigm, the MedTech industry must fundamentally rethink its software development approach, treating it as a scalable, repeatable production system rather than a series of isolated projects tied to individual devices or releases. This transformation requires standardized development environments and toolchains across product lines, enabling reuse of validated software components and common platforms while still accommodating regulatory variations between markets. For example, a shared architecture could support multiple infusion pump models while ensuring compliance with both EU MDR and FDA requirements.

Automation must be embedded end-to-end, covering everything from CI/CD pipelines for continuous integration and deployment to automated verification and validation workflows. Built-in regulatory checks aligned with standards such as IEC 62304, ISO 13485, and regional frameworks like FDA 21 CFR Part 11 should be integrated into these pipelines to ensure compliance is not an afterthought but a continuous process. Unlike automotive, MedTech must place even stronger emphasis on rigorous requirements management, risk control, and design history file (DHF) generation, making traceability from intended use through implementation and clinical validation a core capability of the Software Factory.

To accelerate development and reduce costs, MedTech organizations should also invest in virtualization and digital twins of medical devices. These technologies enable early integration testing and usability assessments without relying on expensive physical prototypes, which are often subject to long lead times and regulatory constraints. For instance, simulating a ventilator's software behavior under different clinical scenarios can uncover issues before hardware is even available.

Finally, cultural change is critical. Development, quality, and regulatory teams must collaborate within a shared factory model, breaking down silos and embedding compliance into every stage of development. This means shifting from late-stage audits to continuous compliance monitoring, supported by automated documentation and reporting tools.

By industrializing software development in this way, MedTech can achieve greater scalability, shorter innovation cycles, and regulatory confidence, enabling the delivery of increasingly software-driven medical devices without compromising patient safety or compliance.

4.4 Benefits of Software Factories for MedTech Companies

Introducing a Software Factory approach enables MedTech companies to dramatically improve development efficiency while strengthening regulatory compliance, an essential capability as software complexity continues to rise. By standardizing development environments and automating key stages of the software lifecycle, organizations can shorten development and release cycles without compromising safety or adherence to standards such as IEC 62304, ISO 13485, and regional regulations like FDA 21 CFR Part 11.

Built-in automation for testing, verification, validation, and documentation generation reduces manual effort and eliminates error-prone activities, resulting in higher and more consistent software quality. For example, automated generation of Design History Files (DHF) and test reports ensures audit readiness and minimizes disruption during regulatory inspections. Continuous traceability from intended use and requirements through implementation, testing, and post-market changes provides full transparency, making regulatory submissions faster and less resource intensive.

The use of reusable software components and platform-based architectures further reduces duplication across product variants and generations, lowering overall development and maintenance costs. At the same time, early integration and virtual testing, enabled through simulation and digital twins, helps identify defects earlier in the development cycle, reducing costly late-stage rework and dependence on physical prototypes. For instance, simulating a surgical robot's control software before hardware availability can uncover critical issues months ahead of traditional testing.

Automated configuration management adds another layer of efficiency by enabling systematic generation of software across multiple configurations, such as regional variants, hardware platforms, and distinct clinical use cases, while ensuring deterministic, compliant, and repeatable results across all versions.

Strategically, Software Factories empower MedTech companies to scale innovation, respond faster to clinical and market needs, and confidently introduce software-driven and connected medical devices, all while maintaining a strong compliance posture and safeguarding patient safety.

4.5 Capgemini's Expertise in Custom Software Factory Solutions

Capgemini brings a wealth of experience and expertise to the design and implementation of bespoke software factories. Our track record demonstrates a deep understanding of the unique challenges faced by various industries and organizations in their software development processes. We excel in guiding clients through the complex transformation journey towards a fully integrated and automated end-to-end Software Factory approach.

4.5.1 Case Study: Scaling Operations with a Software Factory Approach

Capgemini partnered with a French Tier 1 automotive supplier to establish and scale a Software Factory, transforming their software development operations. The project unfolded in three key phases over three years, showcasing the power of the Software Factory model in a real-world context.

The transformation began with a *foundations phase*, where the team shaped the vision, designed the factory operating model, and implemented the organizational structure. This was followed by a *growth phase*, which saw the deployment of embedded SW QMS & processes, implementation of internal toolchains and CI/CD pipelines, and introduction of agile methodologies. The final *scale phase* shifted delivery to best-cost countries, expanded the portfolio to include data & AI products, and industrialized innovation mechanisms.

This transformation yielded significant benefits for the customer. Efficiency improved dramatically, with operations scaling from a small team to over 100 FTEs in just three years, enabling the customer to meet rapidly increasing software demands in the automotive industry. The shift to best-cost countries and leveraging global talent pools resulted in substantial cost savings without compromising quality.

The introduction of agile methodologies and SAFE frameworks greatly enhanced the customer's flexibility and responsiveness to market demands, a crucial advantage in the fast-paced automotive sector. The industrialization of innovation mechanisms accelerated the customer's ability to develop and deploy cutting-edge solutions, including AI and data-driven products, strengthening their competitive position.

Importantly, the Software Factory approach ensured consistent high quality and regulatory compliance throughout the scaling process. This was particularly valuable given the stringent safety and quality requirements in the automotive industry. The implementation of embedded quality management systems and safety processes (ASPICE, Cybersecurity, Safety) provided a robust framework for maintaining standards across all development activities.

This successful transformation exemplifies how Capgemini's Software Factory expertise can drive significant improvements in software development capabilities, operational efficiency, and innovation speed while maintaining strict quality and compliance standards – insights that are highly relevant for MedTech companies facing similar challenges in their software development processes.

4.5.2 Capgemini SDV Factory

Capgemini's SDV Factory presents a comprehensive, cloud-based platform designed to revolutionize the Software-defined Vehicle (SDV) development process. Built on AWS infrastructure and integrating assets from the SDV partner ecosystem, including Capgemini's accelerators, this platform offers an integrated SDV tooling solution that can be rapidly deployed and tailored to individual client needs.

The core vision of our factory approach is to leverage off-the-shelf industrial tooling and apply standardized processes, resulting in an adaptable, repeatable, and efficient solution specifically tailored to SDV use-cases and workflows. This approach enables a shift-left strategy, integrating ecosystem partner tooling, supporting car-to-cloud use-cases in virtual environments, and providing robust OTA, Data Platform, and Fleet Management support. Additionally, our solution incorporates modular GenAI asset integration and offers solution blueprints based on key use-cases.

Our platform is built on five key solution pillars:

1. **SDV architecture & upstream tooling:** This pillar links Application Lifecycle Management (ALM), Requirements Management, Electrical/Electronic Architecture, SDV Software Platform, and Model-Based Systems Engineering (MBSE) workflows. This integration ensures a seamless flow from concept to implementation in the SDV development process.

2. **Virtual engineering lab:** Our solution provides a multi-vendor/target virtual ECU integration environment, linking Software and Verification & Validation (V&V) Factory components. This allows for extensive testing and validation in virtual environments, reducing the need for physical prototypes and accelerating the development cycle.

3. **Software Factory:** This pillar focuses on user workspaces, workflow orchestration, and pipeline management. It provides a collaborative environment for developers, streamlining the software development process and ensuring consistency across teams and projects.

4. **V&V factory:** Our platform incorporates advanced test automation capabilities, including hybrid Hardware-in-the-Loop (HIL) testing. It also supports compliance, certification, and homologation processes, ensuring that developed software meets all necessary regulatory requirements.

5. **Agentic AI framework:** This innovative component is designed to be adaptable and extensible, augmenting and accelerating SDV development through the application of artificial intelligence technologies.

By integrating these pillars into a cohesive platform, our solution offers a comprehensive, efficient, and future-proof approach to SDV development. It addresses the complex challenges of modern automotive software development while providing the flexibility and scalability needed to meet evolving industry demands.

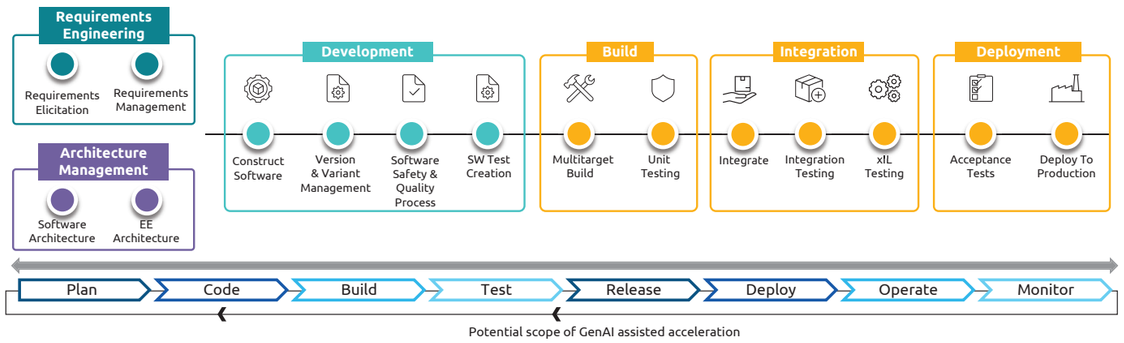


Figure 1 Capgemini SDV Factory

05

Open Source and Generative AI Integration

Open-Source Software Development (OSSD) is becoming increasingly relevant in both automotive and MedTech, but the dynamics differ due to regulatory and safety constraints. This chapter targets to elaborate on how best practices for OSSD from automotive might be applied within MedTech industry.

5.1 OSS Adoption Strategies

Motivation for the OSS strategies adoption

As a key factor for increasing interest for OSSD is current cost reduction push and faster innovation request. These factors are applicable for both industries, though the current landscapes and status quos are different:

Automotive has embraced open source through initiatives like **AUTOSAR, Eclipse SCORE (and other Eclipse projects), OpenADx**, and **ROS (Robot Operating System)** for autonomous driving. Linux-based platforms (e.g., AGL – Automotive Grade Linux) are widely used for infotainment and connectivity. Key focus of these initiatives is development of reusable free of charge non-differentiating components (compilation of reusable assets e.g. middleware stack components, AD applications, OS images, testing tools...), which would enable faster development of core of the new target systems and products and decrease the development costs.

MedTech is more conservative due to **FDA and MDR regulations,** but open source is emerging in areas like:

- **Medical imaging** (e.g., ITK, VTK).
- **Data interoperability** (FHIR standard implementations).
- **AI/ML frameworks for diagnostics** (TensorFlow, PyTorch).

In medical R&D, open-source tools are already often used, but this openness often disappears once research transitions into a regulated product. Also, university hospitals make use of such tool for supporting experimental diagnostics for studies or new treatments.

5.2 Challenges and Constraints

Despite the potential benefits, OSS adoption in MedTech faces unique challenges. Regulatory compliance is paramount, governed by standards such as IEC 62304, FDA regulations, and EU MDR. Every software component, including open-source libraries, must undergo rigorous validation and documentation to ensure safety and reliability. This contrasts with automotive, where ISO 26262 and ASPICE provide structured frameworks for functional safety and process maturity, but allow more flexibility for OSS in non-safety-critical domains.

Intellectual property (IP) protection is another concern. Companies fear that participation in OSS projects may lead to IP leakage or loss of competitive advantage. Automotive has addressed this through consortium-driven governance models, such as those under the Eclipse Foundation, which provide clear licensing terms and neutral collaboration environments. MedTech can adopt similar models to mitigate IP risks and foster trust among stakeholders.

Cybersecurity is a shared challenge. Open-source components can introduce vulnerabilities if not properly managed. Both industries must implement robust security practices, including Software Bill of Materials (SBOM), vulnerability scanning, and secure development lifecycles.

5.3 Best Practices from Automotive OSS for MedTech

Consortium-Driven Governance (Eclipse Model)

Automotive Pattern

Ecosystems under neutral foundations (e.g., Eclipse Foundation projects like Eclipse S-CORE, OpenADx) provide clear contribution rules, IP frameworks, licensing guidance, and conflict resolution – enabling OEMs, Tier-1s, and tool vendors to co-develop non-differentiating components.

MedTech Adaptation

Form a neutral governance consortium with manufacturers, hospitals, research institutes, and tool vendors.

Charter modules by risk class and domain (interoperability, imaging toolkits, compliance tooling).

Provide IP protection mechanisms (inbound/outbound licensing rules, contributor agreements) to mitigate fears of IP loss.

Publish reference architectures and qualification artifacts to reduce compliance burden for adopters.

Eclipse S-CORE as a reference: Use S-CORE's mechanisms for project lifecycle, architecture reviews, and integration tracks as a template for MedTech OSS programs.

Focus on Non-Differentiating Components

Automotive Pattern

AUTOSAR Adaptive, AGL, and Eclipse projects standardize middleware, OS integration, logging, diagnostics, and connectivity layers – enabling OEMs to spend engineering effort on differentiating features (ADAS, UX, energy management).

MedTech Adaptation

Prioritize shared development in lower-risk or non-differentiating domains:

- **Interoperability:** FHIR/HL7 adapters, transport layers, consent management interfaces.
- **Imaging toolchains:** pre-/post-processing utilities (e.g., ITK/VTK extensions), DICOM viewers, anonymization pipelines.
- **Compliance tooling:** SBOM pipelines, automated documentation generators, traceability meta-models.
- **Platform utilities:** logging, monitoring, secure update frameworks, audit trails (non-clinical decision logic).

Outcome

Concentrate proprietary innovation on diagnostic algorithms, device-specific safety logic, and clinical workflow optimizations.

Compliance-Integrated Development

Automotive Pattern

Projects provide qualification kits, documentation templates, safety work products, and automated traceability to support ISO 26262 audits.

MedTech Adaptation

Build OSS components with compliance built-in:

- Include **IEC 62304-aligned** lifecycle documentation (requirements, architecture, V&V reports, risk controls linkage).
- Supply SBOMs, known-issue lists, CVE tracking, patch cadences, and a **secure development lifecycle (SDL)**.
- Provide **validation harnesses** and **reference test suites**, plus guidance for **FDA submissions** (e.g., intended use statements, known limitations, and labeling notes).
- Embed **post-market surveillance hooks:** telemetry schemas, complaint handling interfaces, vulnerability disclosure policies.

Outcome

Transforms OSS from a perceived risk into a compliance accelerator.

Collaborative Test & Simulation Environments

Automotive Pattern

Shared simulation frameworks (e.g., virtual environments for perception, planning, and HIL setups) accelerate integration and validation across suppliers.

MedTech Adaptation

Develop virtual patient and clinical workflow simulation environments:

- Synthetic datasets for imaging modalities and sensor fusion.
- Scenario libraries for device interactions, data flows, and latency considerations.
- Reference performance metrics (accuracy, robustness under noise, failure mode exercises).
- Hooks for clinical risk analysis (hazard injection, mitigations verification).

Outcome

Speeds verification, reduces duplication, and standardizes evidence generation for regulatory submissions.

Responsible Integration of Generative AI

Automotive Pattern

AI supports autonomous features and developer productivity (e.g., test generation, code review, anomaly detection).

MedTech Adaptation

Leverage Generative AI where it does not replace clinical judgment but enhances development and compliance:

- **Documentation automation:** generate trace matrices, test case narratives, risk control rationales, and eQMS logs.uSynthetic data: augment training sets with documented provenance, bias checks, and performance characterizations.
- **Secure coding assistance:** AI-aided static analysis and vulnerability detection for OSS components.
- **Audit readiness:** auto-assemble design history files (DHF), device master records (DMR), and submission annexes from structured repos.

Outcome

Shortens cycles, improves consistency, and retains clinical safety boundaries.

5.4 Generative AI Applications

Automotive software development has embraced GenAI to manage complexity in Software-defined Vehicles (SDVs), autonomous driving systems, and connectivity platforms. These applications require rigorous safety and cybersecurity compliance under ISO 26262 and ASPICE. Similarly, MedTech software development faces IEC 62304, FDA QSR, and EU MDR constraints, making automation and intelligence critical for efficiency without compromising safety.

Automated Test Case Generation

Automotive Pattern

GenAI models generate test cases for functional safety, regression, and edge scenarios, reducing manual effort and improving coverage. Integration with simulation environments (e.g., HIL setups) ensures realistic validation.

MedTech Adaptation

Leverage Generative AI where it does not replace clinical judgment but enhances development and compliance:

- Use GenAI to create **risk-based test suites** aligned with IEC 62304 safety classes (A, B, C).
- Generate **boundary and failure mode tests** for device-software interactions.
- Validate AI-generated tests through **human-in-the-loop review** and **traceability from tests to requirements**.
- Integrate with **virtual patient models** and synthetic datasets for clinical workflow simulation.

Best Practice

Use GenAI to prepare audit-ready documentation and supporting artifacts for formal verification and validation, while ensuring that the actual formal V&V activities themselves remain performed through approved, regulated processes. In general, every agentic AI-based approach should have human-in-the-loop approach for compliancy reasons.

Compliance Validation

Automotive Pattern

AI assists in mapping requirements to safety standards, generating compliance reports, and identifying gaps in ISO 26262 documentation.

MedTech Adaptation

Leverage Generative AI where it does not replace clinical judgment but enhances development and compliance:

- Deploy GenAI to **auto-generate traceability matrices for** linking requirements, design, tests, and risk controls.
- Use AI to **check the completeness** of IEC 62304 lifecycle documentation and FDA submission packages.
- Automate **SBOM analysis** and vulnerability scanning reports for MDR compliance.
- Implement **an explainable AI** for compliance outputs to ensure transparency during audits.

Best Practice

Ensure human validation of AI-generated compliance artifacts and maintain version-controlled evidence repositories.

Code Generation Techniques

Automotive Practice

GenAI accelerates development by generating boilerplate code, configuration files, and even safety-critical logic under strict review processes.

MedTech Adaptation

- Use GenAI for **non-critical code** (UI components, data adapters, logging frameworks).
- For safety-critical logic, apply **restricted code generation** with **formal verification and static analysis**.
- Embed **secure coding guidelines** and **cybersecurity checks** into AI-assisted workflows.
- Maintain **traceability from generated code to validated requirements**.

Best Practice

Adopt **dual-review gates:** automated checks (linting, static analysis) followed by expert review before integration.

5.5 SafeAI

Artificial Intelligence is no longer optional in automotive software development. From autonomous driving systems to predictive maintenance and automated testing, AI is deeply embedded in the Software-defined Vehicle (SDV) paradigm. Similarly, MedTech will inevitably adopt AI for compliance automation, test generation, and clinical decision support. However, **AI introduces new risks –** bias, unpredictability, lack of explainability, and cybersecurity vulnerabilities – that must be managed through **standardized safety frameworks**.

5.5.1 Current Automotive Approach

Automotive safety standards like **ISO 26262** and **ASPICE** were designed for deterministic systems. AI-driven components, especially those based on machine learning, challenge these assumptions because:

- Behavior is **data-dependent**, not fully predictable.
- Verification requires **statistical validation**, not just formal proofs.
- Explainability is limited compared to traditional software.

To address this, initiatives such as **ISO/PAS 8800 (Road Vehicles – Safety of the Intended Functionality for AI) and UNECE WP.29 cybersecurity regulations** are emerging. These aim to define SafeAI principles:

- **Transparency:** AI decisions must be explained.
- **Traceability:** Training data, model versions, and inference logic must be documented.
- **Robustness:** AI must handle edge cases and fail safely.
- **Human Oversight:** Critical decisions require human-in-the-loop validation.

5.5.2 Why MedTech needs a SafeAI Standard

MedTech faces even higher stakes: patient safety and regulatory compliance. Applying AI without a safety framework risks:

- **Regulatory rejection** (FDA, MDR) due to lack of explainability.
- **Clinical harm** from biased or unvalidated models.
- **Cybersecurity breaches** exposing sensitive health data.

Therefore, **SafeAI for MedTech** should:

- Extend IEC 62304 with AI-specific lifecycle requirements.
- Mandate **model validation protocols** (accuracy, bias, robustness).
- Require **audit-ready documentation** for AI-generated artifacts.
- Define **risk-based classification** for AI components (similar to safety classes A/B/C).

5.5.3 Link to Generative AI Applications

The previous section highlighted GenAI use cases – test generation, compliance validation, and code synthesis. These capabilities **cannot be deployed without SafeAI principles**:

- Automated test cases must be **traceable and clinically relevant**.
- Compliance validation outputs must be **explainable and auditable.**
- Code generation must follow **secure coding and verification gates**.

5.5.4 Call for Standardization

Just as ISO 26262 standardized functional safety for automotive, MedTech needs a **SafeAI standard** to:

- Define **acceptable AI practices** in regulated environments.
- Provide **qualification kits** for AI components.
- Enable **regulatory harmonization** across FDA, MDR, and global frameworks.

Outcome

SafeAI is not optional – it is the foundation for trustworthy AI adoption in both automotive and MedTech software development.

From insight to implementation.
Our experts help translate these
principles into real-world results.

Get in touch to continue the conversation.

06

List of Abbreviations

Abbreviation	Definition
AD	Autonomous Driving
ADAS	Advanced Driver Assistance Systems
AGL	Automotive Grade Linux
AI	Artificial Intelligence
ALM	Application Lifecycle Management
ASPICE	Automotive SPICE (Software Process Improvement and Capability Determination)
CFR	Code of Federal Regulations
CI/CD	Continuous Integration / Continuous Deployment
CVE	Common Vulnerabilities and Exposures
DHF	Design History File
DMR	Device Master Record
ECU	Electronic Control Unit
eQMS	Electronic Quality Management System
EU MDR	European Union Medical Device Regulation
FDA	Food and Drug Administration
FHIR	Fast Healthcare Interoperability Resources
FTE	Full-Time Equivalent
FuSa	Functional Safety
GenAI	Generative Artificial Intelligence
HIL	Hardware-in-the-Loop
HL7	Health Level Seven (Interoperability Standard)
IEC 62304	Medical Device Software Lifecycle Standard
ISO 13485	Quality Management Systems for Medical Devices
ISO 26262	Road Vehicles Functional Safety Standard
ISO/PAS 8800	Road Vehicles AI Safety Standard (Safety of the Intended Functionality for AI)
ITK	Insight Toolkit (Medical Imaging)
MBSE	Model-Based Systems Engineering
MedTech	Medical Technology
ML	Machine Learning
OEM	Original Equipment Manufacturer
OpenADx	Open Autonomous Driving Development Platform
OSS	Open Source Software
OSSD	Open Source Software Development
OTA	Over-the-Air (Update)
QMS	Quality Management System
QSR	Quality System Regulation (FDA)
R&D	Research & Development
ROS	Robot Operating System
SAFE	Scaled Agile Framework
SBOM	Software Bill of Materials
S-CORE	(Eclipse) Safe Open Vehicle Core
SDL	Secure Development Lifecycle
SDV	Software-defined Vehicle
SOTIF	Safety of the Intended Functionality
Tier 1	First-level automotive supplier
UNECE WP.29	United Nations Vehicle Cybersecurity Regulation
V&V	Verification & Validation
VTK	Visualization Toolkit (Medical Imaging)

07

About the authors

Tobias Kühnel
Solution Architect

tobias.kuehnel@capgemini.com

Denis Akhmerov
Programm Manager

denis.akhmerov@capgemini.com

About Capgemini

Capgemini is an AI-powered global business and technology transformation partner, delivering tangible business value. We imagine the future of organizations and make it real with AI, technology and people. With our strong heritage of nearly 60 years, we are a responsible and diverse group of over 420,000 team members in more than 50 countries. We deliver end-to-end services and solutions with our deep industry expertise and strong partner ecosystem, leveraging our capabilities across strategy, technology, design, engineering and business operations. The Group reported 2025 global revenues of €22.5 billion.

Make it real.
www.capgemini.com