

SYNTHESIS NOTE

Model-Based Systems Engineering (MBSE): An Imperative for Medical Robot Design

For technical, R&D and quality/regulatory leadership at companies designing and manufacturing robotic medical devices

Octobre 2nd, 2026

Summary

- Medical robotics - robot-assisted surgery, rehabilitation robots, stereotactic neurosurgery, hospital automation - is a fast-growing but structurally complex sector: mechanical, electronic, software and artificial-intelligence (AI) engineering must all come together in a single system, under a dense regulatory framework.
- Recent recalls show that the most critical failures rarely come from one discipline in isolation; they come from incoherent system architecture - broken links between what the system must do for its stakeholders, the functions it performs, and how it is physically built.
- Model-Based Systems Engineering (MBSE) only delivers this coherence when it is grounded in a genuine systems architecture framework - such as the CESAM (CESAMES Systems Architecting Method) architecture framework - applied by a trained system architect, not merely operated as a modeling tool.
- As robotic systems grow more complex and regulation tightens (EU MDR, EU AI Act), companies that keep relying on document-centric engineering - or on MBSE tools used without an architecture framework - face a growing risk of certification delays, non-conformities and recalls.

1. Context: systemic complexity under growing regulatory pressure

1.1 A fast-growing, multidisciplinary sector

Medical robotics today spans a wide range of systems: surgical robots for minimally invasive surgery and orthopedics, image-guided stereotactic neurosurgery platforms, rehabilitation and assistive robots, hospital service robots, and emerging ingestible or implantable micro-robotic systems. The sector is growing fast, but its adoption is still slowed by two factors manufacturers must anticipate from the earliest design stages: the weight of regulatory requirements, and continuing reluctance among patients and clinicians toward these technologies.

The field is inherently multidisciplinary: it combines mechanical, electronic, software, cognitive-science and medical engineering, with each new generation of robot widening the range of expertise that must coexist within a single development program.

1.2 Where failures really come from: broken architecture, not isolated bugs

A modern medical robot contains hundreds of interconnected parts; unexpected behavior is frequently the result of several causes combining - an integration or cross-compatibility issue, a poorly controlled side-effect, a component that is out of tolerance or does not meet the requirements applied to it... - which makes root-cause diagnosis difficult in an organization split into disciplinary silos. Many manufacturers structure their R&D around dedicated centers of excellence (robotics, embedded systems, and so on), which brings deep expertise to each domain but, without a shared architecture description, tends to reproduce fragmented engineering artefacts across teams.

A recent analysis of quality and traceability practice in surgical robotics found that most failures occur at the interfaces between engineering disciplines and between lifecycle phases: mechanical tolerances, the sensor chain, firmware sequencing, algorithms and the clinical user-interaction flow are coupled so tightly that weak traceability causes design baselines to drift. Traceability matrices then frequently fail during audits, because the links they contain are shallow, untyped, or point to artefacts that are no longer under control. In architectural terms, these are breakdowns in the coherence between a system's three core viewpoints - operational (WHY does the system exist? In order to serve which stakeholders does the system exist? To address which needs, use cases, and services?), functional (WHAT is doing the system? What functions does it integrate to provide these services and expected performances?) and constructional (HOW is formed the system? Which subsystems and components are assembled and interfaced with one another to provide these expected functions and performances?) - exactly the coherence that a system architect is responsible for maintaining.

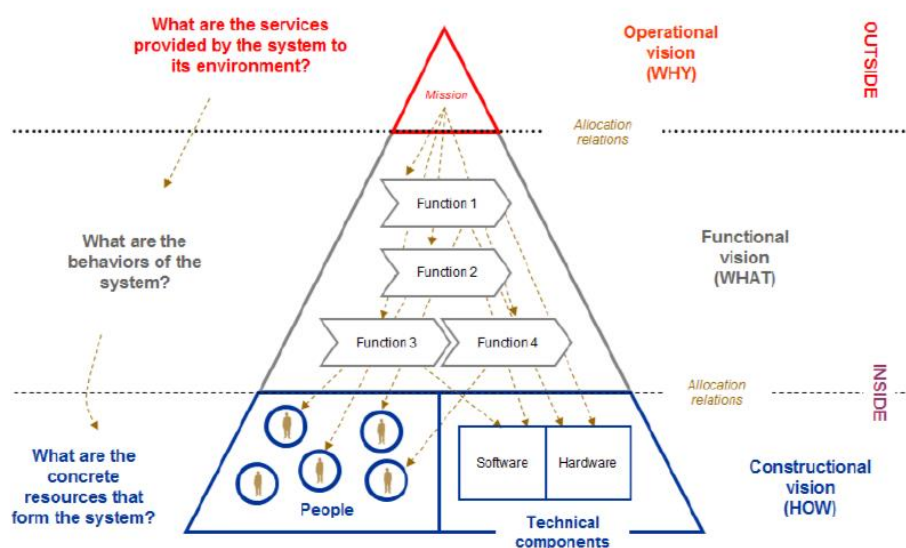


Figure 1 - A medical robot can be effectively made consistent by analyzing and structuring it using the CESAM architectural framework

1.3 A dense regulatory and standards stack

On top of this technical complexity sits a dense body of regulation and standards, which directly shapes architecture and development choices:

Regulation / standard	Scope	Key requirement for a medical robot
EU Medical Device Regulation (MDR, Regulation (EU) 2017/745)	General regulatory framework for medical devices in the EU	Device classification (Rule 11 for software), technical documentation (Annexes I-III), clinical evaluation (Annex XIV)
ISO 13485 (International Organization for Standardization)	Quality management system (QMS)	Control of documentation and design processes throughout the product lifecycle
ISO 14971	Risk management	Reference standard for risk analysis and control, referenced normatively by IEC 62304
IEC 62304 (International Electrotechnical Commission)	Medical device software - software life cycle processes	Harmonized standard (EN 62304); its implementation supports a presumption of conformity with MDR software requirements
IEC 82304-1	Health software product safety	Complements IEC 62304 (process) by covering the safety of the finished software product
IEC 80601-2-77	Particular requirements for Robotically Assisted Surgical Equipment / Systems (RASE/RASS)	Risk management per degree of freedom of the robot; alignment of coordinate reference frames in teleoperation
IEC 62366-1	Usability engineering	Design and evaluation of the robot's human-machine interface
EU Artificial Intelligence Act (2024)	High-risk AI systems	Additional obligations for medical devices embedding AI algorithms (rollout 2025-2027)

This stack is not merely an administrative burden: each normative requirement implies specific engineering artefacts (risk analyses per degree of freedom, software verification evidence, justification of coordinate-frame alignment in teleoperation) that must stay consistent with one another and demonstrable throughout the product's lifetime.

1.4 When architecture breaks down: two illustrations

In 2021, Zimmer Biomet issued a Class I recall (the most severe FDA category) of the cranial application on its Rosa One platform, after a software defect was found that could misread the coordinates sent to the robotic arm during stereotactic procedures, risking mispositioning of the instrument - the second software-related recall to affect this same robotic platform. Separately, Intuitive Surgical recalled its da Vinci SP systems from April 2019 after discovering a software anomaly that could cause unintended vibration of the instrument tips and endoscope, an event the U.S. Food and Drug Administration (FDA) classified as a Class 2 recall in June 2020.

In both cases the immediate cause was labelled "software", but the root cause stems from a lack of consistency in the allocation relationships between different layers of the system's analysis - for example, between the functional requirements defining spatial coordinate management and their proper allocation to different software modules—rather than from a bug confined to a single discipline. but the root cause points to a broken link between architecture viewpoints - for instance, between spatial location functional requirements and their consistent allocation between several software modules - rather than a bug confined to one discipline. It is precisely this kind of inconsistency that a systems-design approach - led cross-functionally by a systems architect - is intended to prevent.

2. Why an architecture-driven MBSE approach solves this

2.1 From documents to a shared architecture model

Model-Based Systems Engineering (MBSE) replaces a patchwork of documents - specifications, spreadsheet requirement matrices, risk analyses kept in separate files - with a structured, machine-readable system model (typically built in SysML, the Systems Modeling Language), which becomes the program's single source of truth. Tools such as Catia Magic (Dassault Systèmes; formerly Cameo Systems Modeler), Capella (originally developed by Thales, now an open-source Eclipse project), IBM Engineering Systems Design Rhapsody, Sparx Systems Enterprise Architect, and PTC Windchill Modeler are commonly used by medical device manufacturers to support this approach, holding requirements, risks and verification evidence inside one traceable environment.

But a tool only ever models what it is told to model. Without a prior architecture description, moving from documents to a model just produces a more sophisticated, tool-based version of the same silos - a risk explored in the next section.

2.2 Architecture comes before modeling: the role of the system architect

This is where an architecture framework such as CESAM matters. It structures in a simple way the description of a system around three coherent viewpoints - operational (interactions with stakeholders and the environment: needs, users, operational scenarios), functional (the functions, modes and flows the system must perform), and constructional (the physical, software and human components that implement them) - and around the system architect, whose role is to keep those three viewpoints aligned and to arbitrate trade-offs across engineering disciplines and lifecycle phases. Modeling only starts once this architectural thinking has taken place; the model is the formal expression of an architecture that a system architect has already reasoned through, not a substitute for that reasoning.

For a manufacturer of robotic medical devices, this means system architecture cannot be treated as an implicit by-product of detailed design. It is a distinct discipline, requiring dedicated system architects who own the coherence of the whole - clinical need, functional behavior, and physical implementation - across mechanical, electronic, software and clinical contributions.

2.3 End-to-end traceability and embedded risk management

Structured this way, system engineering makes V-cycle development truly effective, whether used in traditional project mode or agile mode: a descending branch moving from the operational viewpoint (stakeholder needs and operational scenarios) to the functional viewpoint and then the constructional viewpoint, and an ascending branch of verification - component, integration, system - up to clinical validation. The value of the model lies in horizontal traceability between the two branches: every verification activity is explicitly linked to the definition item whose satisfaction it demonstrates. This is exactly what notified bodies check when reviewing an MDR technical file: that every General Safety and Performance Requirement (GSPR, MDR Annex I) is covered by an identified design, a verification record, and a consistent clinical evaluation (Annex XIV).

Within this same model, the ISO 14971 risk-management process stops being a spreadsheet kept separately by an isolated quality function: every identified hazard is linked directly to system elements, the requirements that control it, and the corresponding verification evidence. This is exactly what IEC 80601-2-77 demands for robotically assisted surgical equipment - risk management per degree of freedom of the robot - a level of granularity that is difficult to sustain coherently without a model that explicitly links each degree of freedom to its risk analysis and requirements, including coordinate-frame alignment, the sensitive point directly illustrated by the recalls in section 1.4.

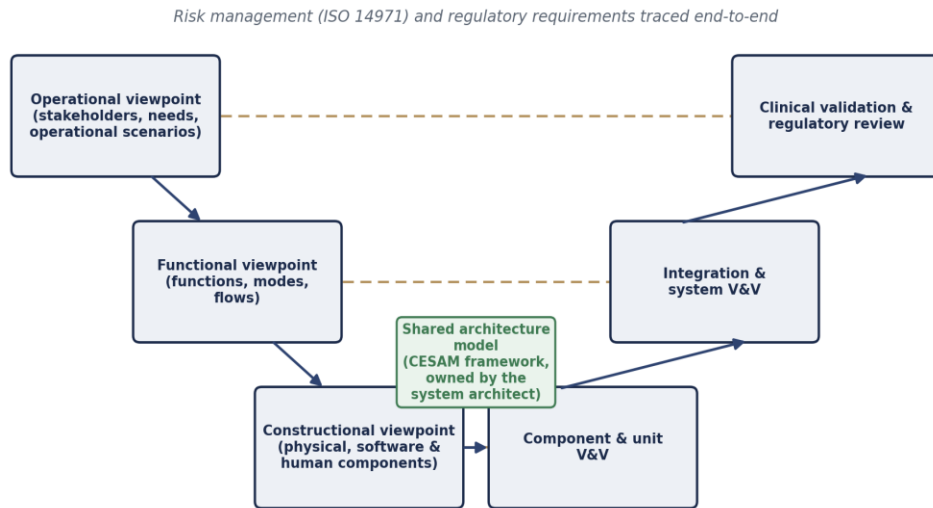


Figure 2 - The CESAM-structured V-cycle: a shared architecture model, owned by the system architect, links every requirement to its verification evidence, with risk management traced end-to-end.

2.4 Collaboration, change control, and an audit-ready technical file

A shared architecture model gives mechanical, electronic and software engineers, regulatory affairs and clinical experts a common language and a common representation, each able to view the product's architecture from their own standpoint without losing overall coherence - reducing the risk that individually strong centers of excellence end up maintaining disconnected artefacts.

IEC 62304's maintenance process requires formal control of software change, and the MDR requires manufacturers to justify whether an update needs a new clinical evaluation. Without a shared model, this impact analysis stays largely manual and error-prone; with one, the impact of a change can be read directly from the traceability links. Combined with product-line engineering, an architecture-framework-based approach also allows validated model libraries and architecture patterns to be reused across national variants of the same robotic platform, as each market imposes its own regulatory and clinical variants.

A well-structured model provides auditable evidence that every requirement has been met and verified, directly usable to build the technical documentation a notified body expects. The chart below illustrates, indicatively, the shift in maturity such an approach can bring on the coherence dimensions that matter most under regulatory audit.

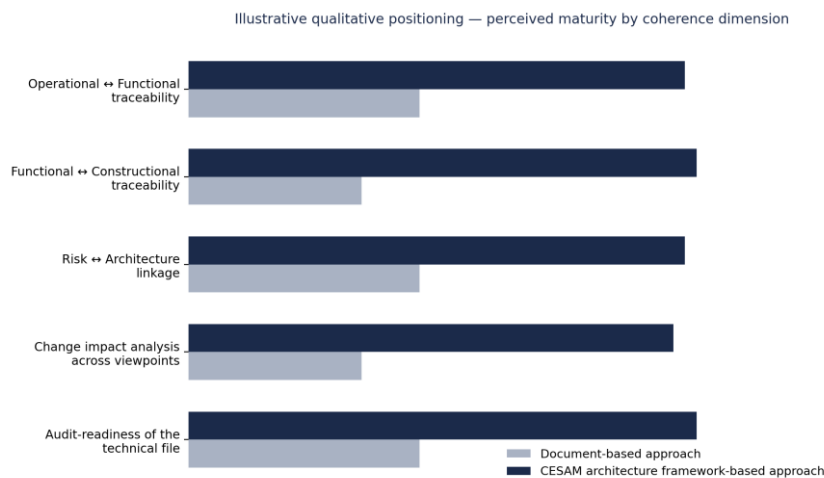


Figure 3 - Illustrative qualitative comparison between a document-based approach and a CESAM architecture framework-based approach, on the coherence dimensions most exposed to regulatory audit.

3. Conditions for success

Success does not depend primarily on tool choice. It depends on having trained system architects, fluent in a structured architecture framework such as CESAM, able to define coherent viewpoints - operational, functional and constructional - before any modeling begins. Companies need a genuine systems-architecture skillset, not just SysML modelers: this typically means investing in architect training and career paths, and clarifying governance over who owns each viewpoint and who arbitrates cross-discipline trade-offs, so the model does not quietly degrade back into disconnected views maintained by each engineering silo.

Beyond people, the architecture model must connect to the existing toolchain - requirements management, Product Lifecycle Management (PLM), the quality management system - to form a genuine digital thread rather than one more disconnected repository. And because the technical file is ultimately judged by regulatory affairs and clinical reviewers, their early involvement in the architecture, not only engineering's, is what allows the model to become the true source of truth for the dossier.

4. Conclusion

Medical robotics combines rising technical complexity - mechanical, electronic, software, AI - with tightening regulation, which the EU AI Act is reinforcing further for algorithm-based devices. In this context, a document-centric approach is becoming structurally unfit for purpose: it multiplies traceability breaks exactly where failures concentrate - discipline interfaces, change control, audit preparation.

A word of caution belongs at the center of this conclusion: deploying SysML modeling tools without a prior architecture discipline simply digitizes the same silos, without correcting their root cause. MBSE is not, first and foremost, a tool - it is a way of thinking through a system's architecture before modeling it. This is precisely the purpose of an architecture framework such as CESAM (CESAMES Systems Architecting Method): it supplies the framework of thought that must precede and structure modeling, carried by a system architect, rather than leaving each team model the system its own way inside a shared tool. Without that prior architectural thinking, investment in an MBSE tool at best relocates the documentation problem into a more sophisticated repository.

Operational recommendation: rather than a generic roll-out, companies should start on one sub-system already identified as critical, or on an upcoming CE-marking dossier, defining an architecture framework sized to the company's needs, appointing system architects to own it, and progressively connecting the model to the existing toolchain (requirements management, PLM, quality management system).

Sources

- CESAMES - CESAM (CESAMES Systems Architecting Method) architecture framework, www.cesames.net
- International Federation of Robotics - Challenges for future health care systems
- AZoRobotics - The Role of Biomedical Engineering in Robotics
- IOPscience - Medical & healthcare robotics: a roadmap for enhanced precision, safety, and efficacy
- Health Tech Exchange - Top Medical Robotics Engineers Share 5 Key Lessons
- Boston Engineering - Accelerating Medical Device Innovation Through Systems Engineering
- Johner Institute - What Manufacturers Need to Know about Robots in MedTech
- UL Solutions - Safety Standards in Healthcare Robotics
- IntuitionLabs - IEC 62304 Guide: Medical Device Software Safety Standard
- PatientGuard - IEC 62304 Explained: Medical Device Software Development Guide
- MedDeviceGuide - IEC 82304-1 Health Software Product Safety
- MGTechSoft - Model-Based Systems Engineering (MBSE) in Medical Device Development
- Springer Nature - MBSE Applied to Design and Development of Biomaterials Devices
- ResearchGate / INCOSE IS 2023 - MBSE in Medical Device Development: Enhancing Efficiency and Quality
- Saratech - What is Model-Based Systems Engineering (MBSE)?
- Lifecycle Insights - The Complete Guide to Model-Based Systems Engineering
- Aras - What Is Model-Based Systems Engineering (MBSE)?
- 321Gang (Medium) - The Powerful Synergy of MBSE and Product Line Engineering in the Medical Device Industry
- Dassault Systèmes - Enhancing Safety and Efficiency in MedTech by Leveraging MBSE
- Enlil - How to Overcome Quality, Traceability, and Compliance Challenges in Surgical Robotics
- Fierce Biotech / Becker's Hospital Review / MD+DI - recall coverage of Zimmer Biomet Rosa One and Intuitive Surgical da Vinci SP