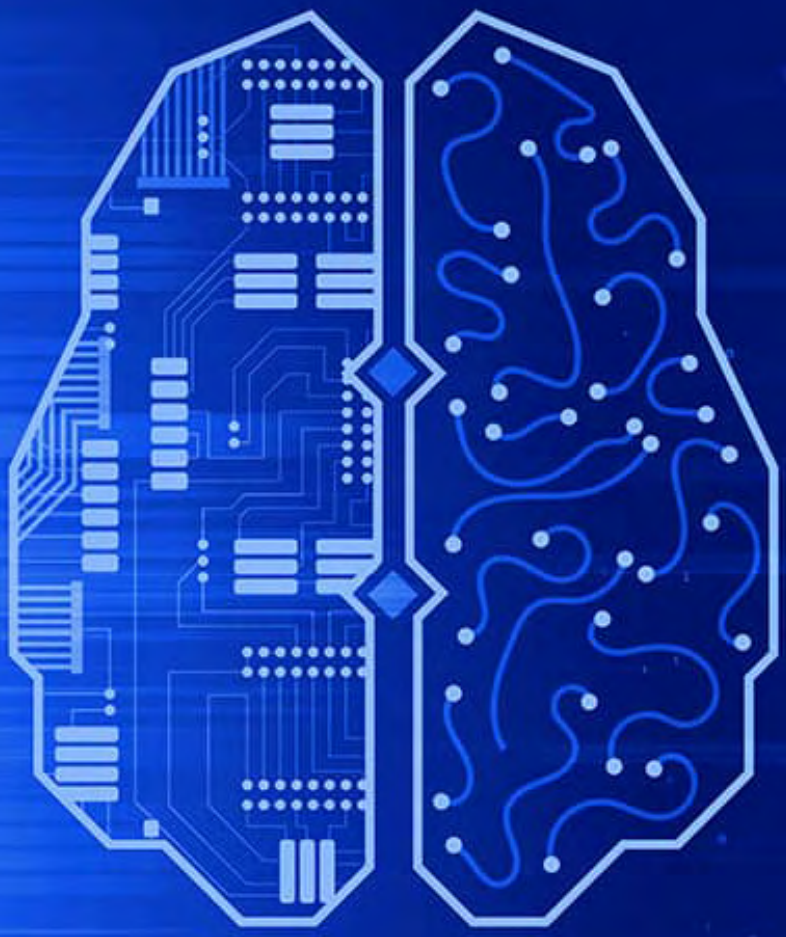
The next Level of Software Development in Regulated Industries enabled by GenAI



Learnings from internal initiatives leveraging GenAI-based solutions for unprecedented implementation efficiency in healthcare & life sciences.

Last updated: 01/2026

zeiss.com/digital-innovation

ZEISS

Seeing beyond

Executive Summary

Generative AI (GenAI) is unlocking step-change efficiency in regulated software development, where various regulations impose rigorous documentation, testing, and audit requirements. Relevant regulations and quality frameworks commonly applied in life sciences, MedTech, and pharma/lab environments include IEC 62304 (medical software lifecycle), the EU Medical Device Regulation (MDR), ISO 13485 (medical device QMS), GxP requirements (e.g., GMP/GLP/GCP), FDA regulations and guidance (e.g., 21 CFR Part 11 for electronic records/signatures), HIPAA for health data protection in the U.S., and the emerging EU AI Act. GenAI can reduce cycle time and cost by automating language- and code-heavy tasks – while keeping full traceability and human review in the loop. Near-term productivity gains of 10–15% of total engineering effort are achievable today, with more than 30% attainable through strategic adoption and scaled deployment. This whitepaper examines how GenAI can enhance software development in regulated environments, with MedTech and life sciences as primary examples, and illustrates its impact through three concrete use cases:

Three ZEISS Digital Innovation initiatives and results:
A Retrieval-Augmented Generation (RAG) initiative demonstrated strong potential to reduce information search overhead and improve user satisfaction. In a large program, engineers reported an average of 60 minutes per day searching across wikis and documents, with low satisfaction (2.7/5). An Azure-based MVP – React frontend, Python/Django backend, Microsoft Entra for secure access – grounds answers in the program wiki and is estimated to implement in 12–16 weeks for €200–300k, with €4–6k monthly operations. Expected monthly savings of €15–24k translate to a conservative break-even in about three years and an optimistic break-even near one year; five-year ROI ranges from 80% to 520%. Success depends on data quality, stakeholder engagement, rigorous validation with golden questions, and alignment with privacy, security, and works council requirements, with clear scaling paths to additional repositories and roles.

A prototype for semi-automated document generation shows how GenAI can compress regulatory document creation from hours or days to minutes through chapter-by-chapter drafting guided by templates and reference examples per document type. Early results indicate roughly 50% automation potential and a modeled saving of about €1 million for 2,000 documents, while improving consistency and accelerating reviews across project management, requirements, and quality roles. Practical lessons include managing token limits via chapter workflows and enforcing precise prompting. The roadmap extends coverage to more document types, dynamic ingestion of legacy content, AI-assisted reviews, risk assessment, automated validation, change control, knowledge reuse, and integrations with requirements and test systems.

An agentic workflow for unit test generation further illustrates GenAI's impact on quality and cost. Using a micro-agent approach with LangChain/LangGraph, the system analyzes existing tests, coverage, and code to propose additional cases that close missed paths, subject to strict quality gates and human review. Across Python, C++, and TypeScript projects, coverage rose substantially – reaching 100% for some classes – while senior developers accepted 85% of generated tests at an average token cost near €0.5 per accepted test. Compared with a similar open-source solution, the approach achieved comparable or better coverage with about 40% fewer input tokens, positioning it for scalable adoption in safety-critical contexts.

Realizing these benefits at scale requires disciplined data preparation and governance, cloud or hybrid architectures with vector databases and secure APIs, multi-agent orchestration, and robust GenAI/MLOps for monitoring, auditability, and drift control. Staged rollouts with clear performance metrics – usage, accuracy, time saved, satisfaction, and adoption – enable systematic tuning. Beyond measurable gains in speed and cost, GenAI elevates consistency, developer satisfaction, and compliance readiness, ultimately accelerating time-to-market and supporting better patient outcomes through safer, more reliable software.

Table of Contents

1. Introduction

1.1 The Need for Innovation in Regulated Software Development

Medical and Life Science software development is inherently complex and faces a multitude of challenges that inhibit innovation and delay time to market. Strict regulatory frameworks – such as FDA, EU MDR, ISO 13485, HIPAA & GDPR, and GxP, 21 CFR Part 11, among other regulations, demand rigorous validation, exhaustive documentation, and ongoing compliance monitoring to ensure the safety, security, and efficacy of medical applications. These requirements often elongate development cycles and drive up costs, placing considerable pressure on teams striving to meet both innovation goals and regulatory expectations.

Compounding this challenge are limitations around high-quality health data. Developers must navigate heightened privacy concerns and data access restrictions, which hinder the iterative development and personalization essential for modern digital health solutions. Unlike conventional, non-critical software, medical systems – such as electronic health records (EHRs), clinical decision support systems (CDSS), and imaging diagnostics – must demonstrate exceptional levels of accuracy, security, and reliability. Failures in these systems can have life-altering consequences, from misdiagnoses to treatment delays or compromised patient data.

The diverse and complex workflows of clinical and laboratory environments present significant challenges for developers, who must create solutions that are not only robust but also adaptable to varying user needs and technical conditions. This complexity necessitates a strategic balance between regulatory compliance, data governance, and agile development practices, ensuring that innovation can thrive without compromising patient safety or data integrity. Addressing these challenges presents a significant opportunity: Innovation is crucial to enhance compliance, reduce costs, and improve patient outcomes through faster, more personalized digital healthcare solutions.

Regulation

Complex regulatory frameworks:

- EU MDR
- ISO 13485
- IEC 62304
- EU AI Act
- …

that require rigorous validation and extensive documentation.

Challenges

Current challenges include:

- High costs
- Slow R&D cycles
- …

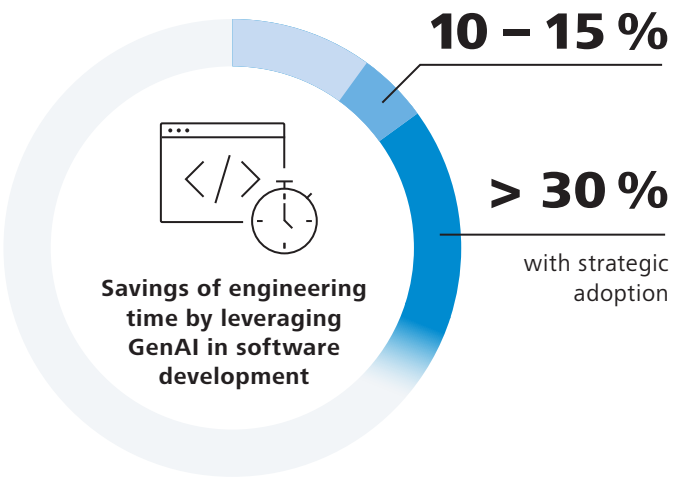
.

Opportunity

Innovation is crucial to enhance compliance, reduce costs, and improve patient outcomes through faster, more personalized digital healthcare solutions.

According to Bain & Company's 2024 report "Beyond Code Generation: More Efficient Software Development", generative AI (GenAI) can save 10–15% of total engineering time today, with the potential to exceed 30% through strategic adoption. This efficiency boost can be realized through targeted use cases, some of which will be discussed in this report. By leveraging GenAI in these domains, R&D teams can simplify compliance processes, reduce time-to-market, and unlock significant innovation potential in healthcare technology.

*bain.com, fda.gov, fda.gov, robinwaite.com, pmc.ncbi.nlm.nih.gov



1.2 What is GenAI?

Generative AI (GenAI) represents a transformative evolution within artificial intelligence, characterized by its ability to create new, original content – ranging from human-like text to detailed images, audio, and similar content like program source code, scripts, and diagrams – by learning intricate patterns from enormous datasets. Unlike traditional AI or conventional machine learning, which primarily focus on recognizing patterns and making predictions based on predefined rules or labeled data, GenAI employs advanced deep learning architectures such as large language models (LLMs), diffusion models and generative adversarial networks (GANs) to generate novel outputs that can mimic human creativity and reasoning. A key enabler of this shift is GenAI's ability to "understand" and interpret vast amounts of source data in a more organic, context-aware manner. This capability unlocks significantly broader use cases, particularly in processing and synthesizing large volumes of unstructured human- and machine-generated data and information.

mckinsey.com, techtarget.com

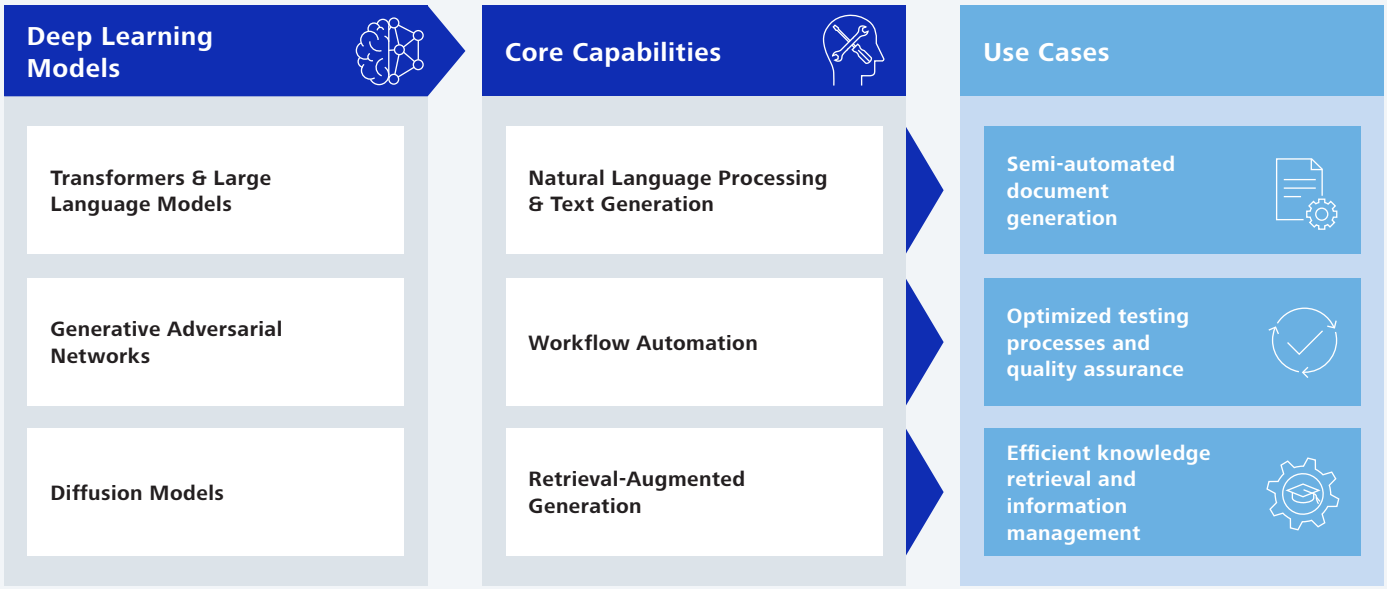
1.3 Relevance and Opportunity for Healthcare & Life Sciences

Generative AI is rapidly emerging as a pivotal technology in software development for Health and Life Sciences, offering targeted capabilities that address longstanding bottlenecks across design, development, validation, and compliance. Rather than simply enhancing healthcare and life science workflows, GenAI can directly impact core engineering tasks via its core capabilities like image and text creation – especially within a highly regulated environment.

By leveraging GenAI, R&D teams can semi-automate the generation of regulatory documentation, which significantly reduces the manual effort required to meet standards such as FDA, EU MDR, GMP, ISO 13485, and GxP, 21 CFR Part 11. It also streamlines the retrieval and summarization of complex technical information from vast, structured and unstructured datasets, improving knowledge management and traceability. Furthermore, GenAI accelerates software testing cycles through the automated generation of test cases, synthetic data, and model-based validation scenarios – without compromising data privacy. These use cases not only enhance productivity but also support faster time-to-compliance, higher software reliability, and ultimately better patient safety and outcomes.

rishabhsoft.com, arxiv.org, persistent.com, techtarget.com

2. Overview of GenAI Technologies

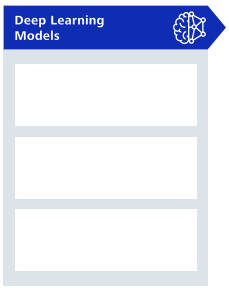


2.1 Foundational Models & Techniques

Foundational models and techniques in generative AI have fundamentally reshaped the technological landscape, enabling breakthrough applications in healthcare and life sciences. Large Language Models (LLMs), such as GPT4, have demonstrated remarkable proficiency in understanding and generating human-like text and structured code, underpinning advanced applications from automated clinical and laboratory documentation to patient communication tools.

In parallel, diffusion models and Generative Adversarial Networks (GANs) have emerged as powerful engines for image generation, enabling the creation of well-structured diagrams and graphics as well as synthetic medical images that both augment training datasets and enhance diagnostic accuracy.

Underlying these innovations are transformer architectures, which facilitate robust sequence-to-sequence learning for context-aware data processing – capabilities essential for tasks ranging from clinical data analysis to the automation of complex workflows.



reuters.com, persistent.com, techtarget.com, xcubelabs.com

2.2 Key Capabilities and Use Cases enabled by GenAI

Generative AI is transforming healthcare and life sciences by streamlining operations, accelerating innovation, and enabling personalized care. Built on powerful capabilities like natural language processing, computer vision, code generation, synthetic data modeling, and workflow automation, GenAI addresses key pain points across clinical, laboratory, and research domains.

- **Streamlined Clinical and Laboratory Documentation:** GenAI automates the creation of patient notes, lab reports, discharge summaries, and clinical trial documentation using NLP models that transform unstructured data into structured, standardized records. In laboratory settings, it supports documentation of experimental protocols, lab results, and quality reports – reducing manual input while improving traceability, compliance, and reporting accuracy.

- **Advanced Diagnostics and Decision Support:** In radiology, pathology, and laboratory medicine, GenAI-powered computer vision systems enhance image analysis by improving resolution, detecting anomalies, and supporting risk stratification. These capabilities enable earlier and more precise diagnostics in both clinical and research settings, such as identifying cellular morphologies or analyzing histological slides.

- **Synthetic Data Generation for Research, Development, and Training:** GenAI generates privacy-preserving, high-fidelity synthetic patient or biological datasets that reflect real-world patterns. These datasets are essential for training AI models, simulating rare conditions, augmenting clinical trials, and safely conducting virtual experiments. In laboratories, synthetic data can simulate assay results or experimental conditions, supporting more robust validation and reproducibility.

- **Text & Code Generation and Engineering Automation:** GenAI assists developers and lab engineers by generating documentation, software prototypes, test scripts, laboratory automation code, and data processing pipelines from natural language prompts. This capability accelerates digitalization across regulated environments, such as LIMS (Laboratory Information Management Systems) and electronic lab notebooks.

- **Workflow Automation Powered by Multi-Agent Systems:** Behind the scenes, AI agents collaborate to automate complex processes across healthcare and life sciences – ranging from regulatory documentation and audit trail creation to sample tracking and experiment scheduling. These multi-agent systems enable adaptive workflow orchestration that improves speed, reduces errors, and ensures regulatory compliance.

- **Efficient Information Management with RAG:** Retrieval-Augmented Generation (RAG) enhances GenAI by grounding outputs in trusted sources like QMS templates, regulatory guidelines, and clinical databases – enabling compliant, customized content generation. Combined with intelligent search and summarization, it streamlines information retrieval, reduces review time, and supports faster, audit-ready documentation – modernizing QMS and decision-making in healthcare and life sciences.

With these capabilities, generative AI is not just optimizing isolated tasks – it's reshaping the foundations of how healthcare providers, life sciences researchers, and laboratory professionals operate. From streamlining documentation to accelerating drug discovery, GenAI enables a smarter, faster, and more personalized future for science and care delivery.

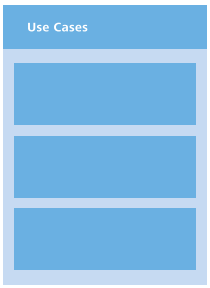
techtarget.com, c3.ai, reuters.com, xcubelabs.com, compunnel.com, persistent.com

2.3 Examples & Impact of GenAI-based Solutions

The vision for leveraging generative AI in medical software is centered on transforming healthcare delivery by enhancing diagnostic tools and accelerating research and development cycles. By integrating AI into clinical workflows, organizations aim to develop more precise diagnostic algorithms that can analyze imaging and genetic data with unprecedented accuracy – leading to earlier detection of diseases and more effective interventions. This strategy not only targets improved patient outcomes but also strives to reduce overall treatment costs and streamline operations.

persistent.com, c3.ai

Generative AI has the potential to drastically accelerate R&D cycles in medical software development. By automating routine coding tasks, generating regulatory documentation, and enabling precise, context-aware knowledge retrieval from internal systems, GenAI reduces the time and effort required across the entire development lifecycle. These capabilities support faster prototyping, more efficient test case generation, and streamlined compliance processes – key bottlenecks in traditional medical software workflows. When integrated responsibly within regulatory frameworks and data privacy requirements, GenAI empowers development teams to innovate faster, reduce time-to-market, and focus more on high-value activities like safety optimization, clinical relevance, and user-centric design.



To advance medical software development, we initiated targeted efforts in three key areas: **Semi-automated regulatory document generation**, optimized testing processes and QA with **automated unit test case generation**, and efficient knowledge retrieval and information management with **Retrieval-Augmented Generation (RAG)** for improved knowledge retrieval. The following sections present in detail the comprehensive assessment of RAG's potential impact as a basis for future investment decisions and the implemented MVPs for documentation and test generation including preliminary results. Together, these initiatives aim to streamline development workflows and accelerate innovation while maintaining compliance and quality.

3. Use Cases

3.1 Use Case 1: Advanced Knowledge Retrieval and Preparation with Retrieval-Augmented Generation (RAG)

Context

In many medical software programs following IEC62304, the vast amounts of documentation and expansive wikis represent immense knowledge assets, yet finding the right piece of information remains a challenge. Technical teams often spend excessive time – sometimes up to several hours per week – sifting through disjointed resources like API documentation, sprint reviews, or architectural details. While current search methods provide access to this wealth of data, there is significant potential to further enhance efficiency and user experience. Conventional LLM-based chatbots rely solely on pre-trained models and, as a result, cannot tap into the rich, up-to-date company-internal repositories maintained within these regulated environments.

A RAG-enabled solution leverages internal data by processing extensive documentation into embeddings – mathematical representations capturing the semantic meaning – and storing them in a vector database for rapid retrieval. The RAG approach then follows a three-step process: First, in the retrieval phase, the system swiftly identifies the most relevant documents from this vector database; next, during the augmentation phase, it supplements these findings with additional context to ensure the information is tailored to the user's query; and finally, in the generation phase, the language model combines this context with its inherent knowledge to deliver precise, context-specific responses. This powerful integration promises to streamline workflows, reduce time spent searching for critical information, and boost overall efficiency, making it a highly valuable enhancement for medical software programs with expansive knowledge bases operating under standards like IEC62304.

sap.com, aws.amazon.com, medium.com

The current information retrieval solutions, such as search bars, are inadequate, evidenced by a low satisfaction rating of 2.7 out of 5 stars for an exemplary wiki. Employees spend an average of 60 minutes per week searching for relevant information, highlighting inefficiencies. There is a clear need for improved methods, with a preference for implementing RAG-enabled chatbots to enhance information retrieval processes (source: Internal analysis).

Average time spent retrieving information in wikis per week per person:

60 min

How satisfied team members are with search results in wikis:
1 very low – 5 very high satisfaction

Average Rating

2.73

Solution Approach

We assessed the potential of a RAG solution for a large internal software program developing infrastructure according to IEC 62304 to unlock further efficiencies in handling vast amounts of documentation and extensive wikis. In our approach, we started with an in-depth assessment of existing knowledge bases by clearly cataloging and evaluating the most critical repositories, such as the developers' wiki and the document management system. Recognizing the developers' wiki as a strategic starting point for a Minimum Viable Product (MVP) RAG system, we aligned internally around the project scope and defined a specific target group covering functions like DevOps, Testing, Development, and Software Architecture. We then conducted surveys, covering roughly 10% of this target group, to quantify the time invested in information retrieval and the associated levels of user satisfaction. This survey provided a concrete baseline to measure expected improvements.

Subsequently, we performed a thorough assessment of the implementation effort by delineating the necessary roles, work packages, and estimated project duration. This phase also examined operational and maintenance costs, including platform consumption, service team expenses, and costs related to aligning with IT demand processes and workers council requirements. An initial solution architecture was developed in collaboration with partners, allowing us to define a cost range, estimate potential time savings, and ultimately calculate the Return on Investment (ROI) as well as the break-even point for the RAG solution. In the next stage, we explored various integration scenarios where a RAG-enabled chatbot could be deployed, while reassessing and validating the suitability of our primary data source – the developers' wiki. We also deliberated on scaling opportunities beyond the MVP and even beyond the immediate project, ensuring the solution could adapt to future demands.

Finally, we focused on selected teams with the highest potential for efficiency gains: We defined key use cases for information retrieval and conducted follow-up internal surveys to measure current resource investment, satisfaction, and overall sentiment towards chatbot integration. A parallel evaluation for an external use case provided additional validation of our assumptions, with all findings comprehensively presented to management. This multi-stage, data-driven approach not only solidified the feasibility and strategic value of a RAG system but also set the foundation for leveraging GenAI to substantially improve efficiency and user satisfaction in medical software development.

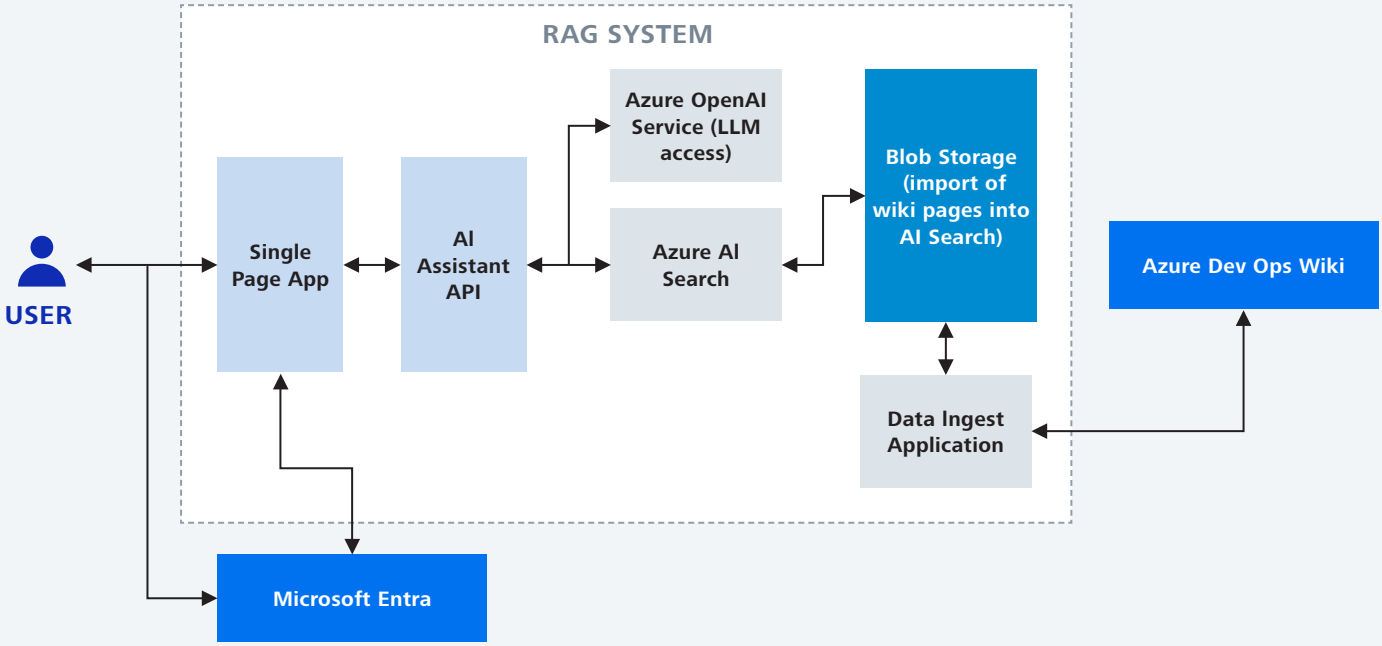


Figure 1 Solution design of the RAG System MVP (simplified)

Following the analysis, we implemented a minimum viable product (MVP) of the RAG solution using the Azure ecosystem (see Figure 1). The RAG System MVP is a user-friendly chat interface designed to assist users by generating AI-powered answers augmented with content from a program-level wiki. It consists of a React-based Single Page Application for user interaction and a Python/Django REST framework backend that interfaces with Azure AI Services for completion generation. The system ingests data from the wiki stored in Azure DevOps, ensuring relevant responses, and utilizes Microsoft Entra for secure user authentication. Accessible via a web browser, the system facilitates knowledge acquisition and insights into the program ecosystem, while operators are provided with comprehensive documentation for deployment and maintenance using Azure Cloud services.

Results

Our evaluation began with an initial survey of technical roles, revealing that program members spend an average of 60 minutes per day searching for information, though individual times ranged widely from as little as 5 minutes to as much as 300 minutes. The information they sought included API documentation, sprint reviews, architectural details, and similar materials. With an average satisfaction score of 2.7 out of 5, respondents expressed considerable dissatisfaction with the current search processes, and the majority indicated that a chatbot would be beneficial. In fact, many participants stated they would prefer to interact with a RAG-enabled chatbot for information retrieval rather than rely on asking colleagues directly, clearly highlighting an opportunity for improvement.

Regarding implementation efforts, our assessment identified that deploying a RAG pipeline is streamlined on cloud platforms like Azure, which offer ready-made services to support key functions. The primary implementation tasks include preparing the data from the designated knowledge base – this involves generating embeddings and creating a vector database – setting up and deploying the data pipeline, and configuring system prompts for the LLM. Additionally, a manual validation process is required, involving the development of a set of golden questions and answers in collaboration with domain experts.

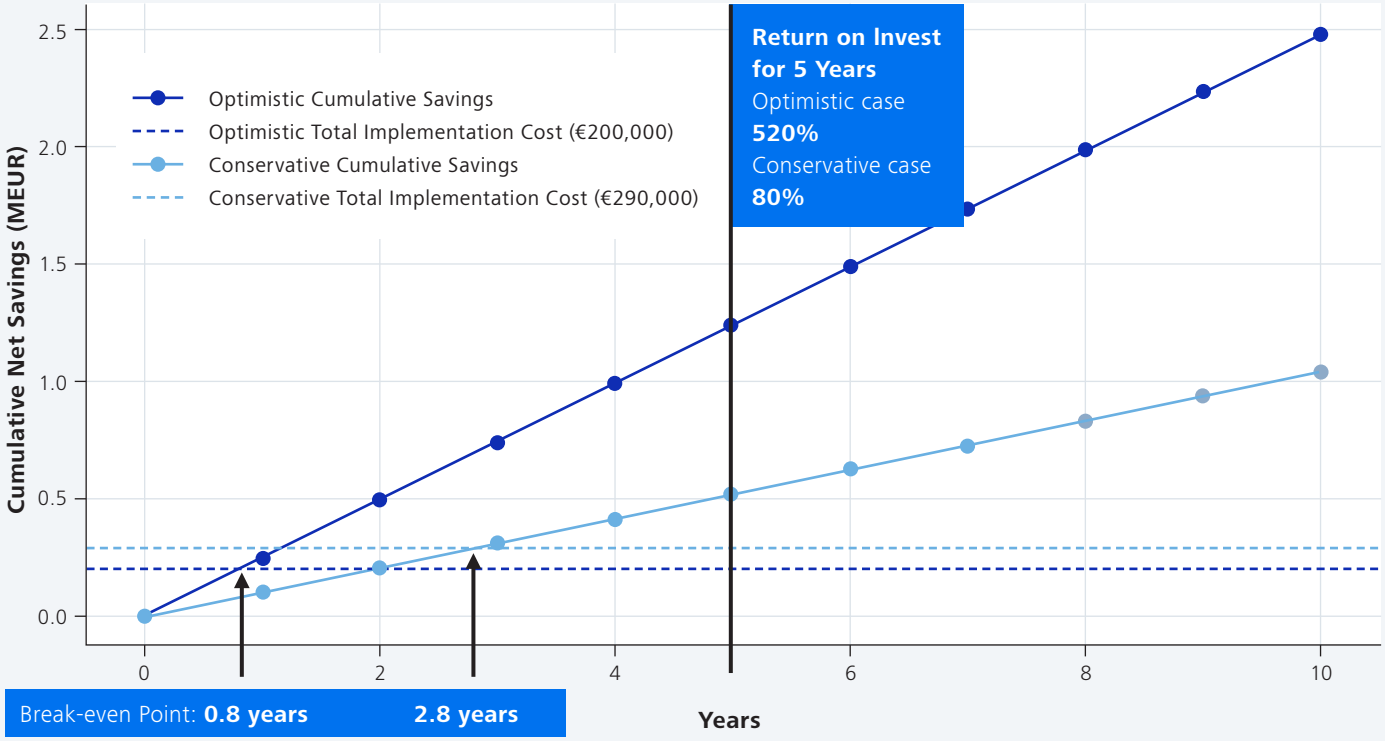


Figure 2 ROI analysis with net savings over time for the RAG System MVP

The questions are run through the system to compare outputs against desired responses, leading to adjustments in system prompts or refinements in the source documentation. Although advanced options such as fine-tuning the LLM or introducing knowledge graphs were noted, they were not pursued at this stage. Supplementary activities like documentation and integrating a frontend were also factored into the overall effort. Overall, the implementation is estimated to take between 12 and 16 weeks, with team costs ranging from about 200k€ (optimistic) to 290k€ (conservative) and additional monthly maintenance and operations costs of approximately 4k€ to 6k€, depending on service consumption and maintenance requirements. Cost-saving estimates were derived from the expected time savings in searching and retrieving information per program member based on their hourly rates and projected efficiency improvements of 30% (conservative) to 50% (optimistic) when using the RAG system. This leads to estimated monthly savings of roughly 15k€ in the conservative case and 24k€ in the optimistic case. These improvements translate into a break-even period of approximately 2.8 years and an 80% ROI over 5 years under conservative assumptions, while the optimistic scenario shows a break-even of just 0.8 years with a 520% ROI after 5 years, indicating significant potential (Figure 2).

In parallel, we conducted a detailed data quality assessment of the developers' wiki by analyzing factors such as the volume of textual and unstructured data (images, diagrams), and consulting experts on issues like duplicated or outdated information, update regularity, and overall structural definition. The conclusions confirmed that the wiki is a suitable repository for the RAG system, although minor adaptations may be necessary during system tuning. For integration, we identified an existing company web frontend as the target, which promises ease of integration and centralized accessibility as well as options for user authentication and authorization. Scalability was also a key focus. We discussed extending the RAG system beyond the wiki to encompass other vital knowledge sources like the document management system, as well as reaching additional target groups such as nontechnical roles (e.g., Project Leads, Product Owners). With all knowledge bases integrated, deployment could expand to other programs within the organization, with anticipated implementation cost savings of 30–40% due to already established data pipelines and internal expertise. However, each new use case would still require steps like in-depth assessments, data preparation and system validation.

Lastly, we conducted detailed evaluations in two relevant projects within the program to define key use cases for information retrieval. These crucial use cases were first knowledge transfer for new teams and second onboarding for client application teams, utilizing the infrastructure developed in the program as the backend for their applications; here, time savings of roughly 25 to 50 minutes per person per week were expected, with a user satisfaction score of around 3.3 out of 5 – indicating clear room for improvement. The third use case focused on software development based on existing software components, where developers and Product Owners currently invest significant time reviewing existing documentation to build their solutions based on the existing components. In this context, our research indicated that a RAG system could save an estimated 70 to 110 minutes per week per person, with user satisfaction being notably low at 2.5 out of 5, thereby reinforcing the potential value of a RAG-enabled chatbot to support efficient information retrieval and improved user satisfaction.

Overall, the comprehensive evaluation – from initial survey results through implementation feasibility, cost-benefit analysis, data quality assessments, integration and scalability planning, to detailed project-specific use case evaluations – demonstrates the significant potential of a RAG system to enhance information retrieval, reduce wasted time, and ultimately improve user satisfaction across multiple domains within medical software development programs.

Lessons Learned

Our comprehensive evaluation of implementing a RAG-enabled chatbot in a medical software development environment has yielded several key lessons. Firstly, conducting an in-depth assessment is essential to understand the unique potential of a RAG solution for each specific use case. It is crucial to obtain data directly from the relevant context, such as the specific program where the solution will be leveraged, to accurately assess the current situation regarding time invested in information retrieval and user satisfaction. This helps validate the feasibility, desirability and viability of a RAG solution in that context. Engaging the target audience early on ensures that the solution developed will be utilized effectively, avoiding the pitfall of building a system that remains unused.

Additionally, while financial assessments showed potential cost savings from implementing a RAG solution, it is important to recognize that the primary value lies in its ability to enhance user satisfaction. A RAG solution can significantly improve the experience of employees by providing timely and relevant information, thereby enabling a modern tooling environment that empowers them to perform their tasks more efficiently. This increased satisfaction can lead to higher employee morale and retention, which are critical factors in a competitive industry.

Moreover, the cost savings associated with a RAG solution are often indirect. Employees are not automatically compensated less simply because they save time; rather, the time they save can be redirected towards more value-creating tasks. For instance, developers can focus on feature implementation in software projects, allowing development initiatives to progress more rapidly. This shift in focus not only accelerates project timelines but may also enhance the overall quality of the software being developed, as employees can dedicate their efforts to innovation and improvement rather than mundane information retrieval tasks.

From a technical standpoint, the hurdles in setting up a RAG pipeline are relatively low, thanks to streamlined implementation processes provided by major platform providers and ready-made services. The quality of data in the connected knowledge bases is crucial. If the data is outdated or poorly maintained, the RAG system will not deliver accurate information. Therefore, establishing a solid data foundation is necessary, although it is advisable not to overinvest if reliable knowledge bases like wikis or document management systems are already in place. Conducting an initial assessment, setting up an MVP, and evaluating its performance can provide valuable insights into any necessary adaptations in the knowledge base.

Finally, considerations around data privacy, alignment with internal IT processes, cybersecurity, and compliance with workers councils (e.g., for saving chat history) are critical to successfully deploy a RAG system. These aspects ensure that the solution not only meets technical and user requirements but also adheres to organizational policies and regulations. By addressing these factors, organizations can maximize the benefits of a RAG-enabled chatbot, ultimately leading to a more efficient, satisfied, and productive workforce.

Next Steps

Following the assessment and the successful setup of the MVP, the next step is to deploy the MVP solution and initiate its rollout to the defined target group of technical specialists within the program. A staged rollout is recommended – starting with a select group of users to gather early feedback, validate the solution's effectiveness in real-world use, and perform initial system tuning.

The first phase of tuning will focus on adjusting system prompts, refining retrieval logic, and curating or enriching the underlying knowledge base to improve response quality and relevance. Should these adjustments prove insufficient in meeting user expectations or delivering the desired efficiency gains, we will evaluate the adoption of more advanced RAG architectures such as Graph-RAG for enhanced contextual linking or Agentic RAG to enable more dynamic, multi-step reasoning and task execution. These approaches offer potential for greater accuracy, better information synthesis, and a more interactive user experience. Once the system is delivering the required performance, the rollout can be extended to the entire target group.

During this phase, it is essential to measure a variety of relevant metrics to comprehensively assess the impact of the RAG system. These metrics could include:

- User interactions per week with the system: Tracking how frequently users engage with the chatbot.
- Average time spent on information retrieval: Measuring the time users spend searching for information using the RAG system.
- User satisfaction scores: Conducting follow-up surveys and interviews to gauge user satisfaction and perceived usefulness of the system.
- Accuracy and relevance of responses: Evaluating the quality of the information provided by the chatbot.
- Efficiency improvements: Calculating time savings and productivity gains based on user feedback.
- Adoption rate: Monitoring the percentage of the target group actively using the system.

Collecting and analyzing these metrics after the system has been in use for several weeks will provide a comprehensive view of the system's effectiveness and its real value. This data will be crucial for making informed decisions regarding further scaling and potential enhancements of the RAG system.

3.2 Use Case 2: Semi-Automated Document Generation

Context

Language-related tasks like documentation make up 62% of employees' total working time, and 65% of that can be optimized through augmentation and automation with GenAI.

Accenture.com

Especially in software engineering for medical domains significant documentation is necessary for later successful certification. An exemplary documentation process at ZEISS, essential for compliance, involves the creation of approximately 125 documents with about 30 to over 100 pages in size across 6 phases and 7 gates. This time-consuming process includes multiple review cycles to ensure quality, making it indispensable and unavoidable.

Documents
125

Gates
7

Pages per document
30 – 100

Phases
6

Generative AI can significantly enhance and automate the process of document generation across various domains. We have created an internal prototype of the so-called Chat based genAI- DOCumentation creator tool (CAIDOC) that is focused to assist software project teams to fulfill their documentation needs. CAIDOC reduces the burden of monotonous documentation creation and adaption tasks, which increases the efficiency of the documentation process and shortens documentation review and update cycles. CAIDOC can also be tailored and trained for different document templates, documentation cycles and processes, as well as adapted to requirements of different industries or business contexts via Retrieval-Augmented Generation approaches. The cost saving opportunity from solutions like CAIDOC is estimated to be about 1 million € considering the completion of 2000 documents[1].

In highly regulated domains such as healthcare, legal, or government, software development demands the creation of extensive and formalized documentation. This often involves populating complex templates with recurring project- and product-specific information, which must be presented from multiple perspectives across various document types. As a result, teams are required to rephrase and reformat the same core content repeatedly – making the process time-consuming, error-prone, and discouraging, especially for those less experienced with regulatory documentation standards.

The documentation burden is further compounded by unstructured review processes, where quality and completeness depend heavily on the reviewer's individual interpretation. This often leads to repeated revision cycles and inconsistent outcomes.

1 Considering 50% documentation automation and assuming 5 person-days per document on average to draft, review, refine and complete.

With our CAIDOC prototype, we demonstrated that initial drafts of regulatory documents could be generated within minutes rather than hours or days. This represents a significant reduction in manual effort, and a 50% automation potential may be considered a conservative estimate based on early results. By reducing the cognitive and operational load of documentation, such solutions can free up valuable expert time and improve consistency across document sets.

Solution Approach

AI technologies can meaningfully reduce the documentation burden in medical software development by learning from existing project artifacts. By analyzing historical documents, AI can assist in creating new, compliant drafts more quickly and consistently – improving both the speed and quality of documentation across the software lifecycle.

Our **CAIDOC** prototype exemplifies this potential by supporting users in generating documentation based on existing examples and project-specific input. To use CAIDOC effectively, each document type requires a set of at least five references:

One blank template to guide generation, and four finalized documents from previous projects to provide context. Once ingested, the system enables chapter-by-chapter drafting through a web-based interface. Users select the document type (e.g., Software Development Plan or Release Information), enter basic project metadata, and are guided through intelligent, context-aware suggestions. Once completed, the full document can be exported in .docx format in line with predefined templates.

As shown in the solution design in Figure 3, CAIDOC begins with the user defining the project context. This is followed by interactive chapter-based generation using pretrained models and contextual examples from the knowledge base. The system leverages AI reasoning and previously ingested data to populate templates, validate structure, check for correctness & completeness and produce a high-quality draft document tailored to organizational standards. After formatting and exporting the final document is ready for review by a quality manager before being integrated into the document management system.

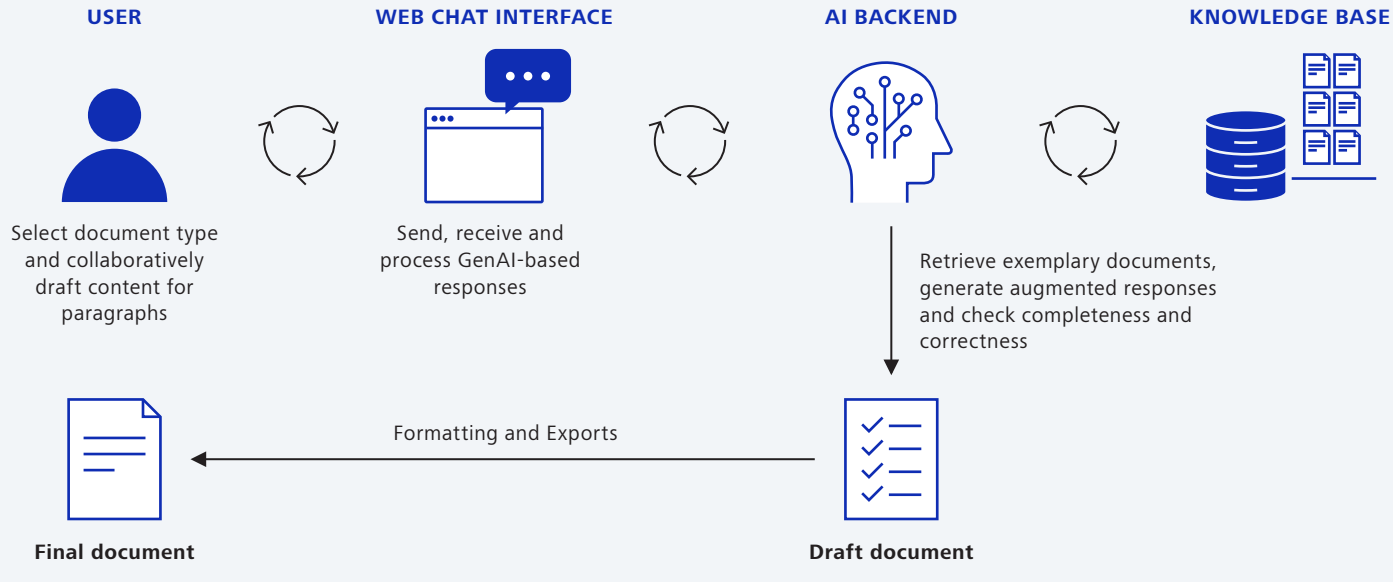


Figure 3 Solution design of CAIDOC (simplified)

CAIDOC is particularly useful for roles such as Software Project Managers, Requirements Engineers, Quality Managers, and Business Analysts – especially those working in regulated environments. Looking ahead, CAIDOC aims to expand to other units within ZEISS, supporting documentation tasks in sectors such as healthcare, legal and compliance, cybersecurity, and public administration – anywhere templated, rule-driven documentation is a core requirement.

CAIDOC is implemented using a robust tech stack comprising Java, React, Langchain, and Postgres with embeddings, alongside Websocket for real-time communication. This architecture supports a multi-AI agent approach, where the Main Assistant acts as the message orchestrator, coordinating interactions between various specialized agents. These include the Template Expert AI Agent and the Document Expert AI Agent, each designed to handle specific tasks within the document creation process.

The system incorporates several tool agents to enhance functionality:

- **SectionListTool:** Manages chapters during the creation and ingestion process.
- **SectionTopicsTool:** Handles the content of individual chapters.
- **SectionExampleTool:** Provides examples for specific chapters.
- **ConfirmSectionTool:** Saves content for generation upon user agreement.
- **GenerateDocumentTool:** Produces the complete .docx document.
- **SectionQueryTool:** Retrieves previously generated sections, allowing continuation of document generation if paused.

A notable feature of CAIDOC is its approach to memory management. While there is no persistent chat memory, the SectionQueryTool can access already generated chapters, ensuring continuity in document creation without re-feeding the chatbot once closed. Additionally, every ingested and generated chapter undergoes a similarity search across different agent aspects, ensuring consistency and relevance in the content produced. This sophisticated integration of AI agents and tools within the system enables CAIDOC to deliver efficient and personalized document generation solutions.

Lessons Learned

Throughout the development of CAIDOC, we encountered several challenges that led to valuable insights and improvements. One significant limitation was the context-window token limit for longer documents. To address this, we devised a method to generate documents chapter-by-chapter. This approach ensures that both the ingestion and generation processes are manageable and efficient, allowing us to handle extensive documentation without exceeding token limits.

Additionally, we found that for more reliable document generation, it is essential to ingest at least four example documents. This provides the AI with a robust dataset to reference, resulting in higher quality and consistency in the generated documents.

Another critical lesson was the importance of providing specific instructions and limitations in the prompts. For instance, directives such as "don't mix the chapters in the answer" were necessary to prevent the AI from producing unreliable outputs. By incorporating these detailed guidelines, we were able to significantly improve the accuracy and relevance of the generated documentation.

Next Steps

As CAIDOC evolves, several enhancements are planned to broaden its capabilities and streamline documentation processes for the Software Development Lifecycle (SDLC). The following steps outline the key enhancements in the backlog:

1. **Expanding Document Types:** CAIDOC will support a wider range of document types, including agile artifacts such as user stories and sprint retrospectives. This expansion aims to cater to diverse project methodologies, ensuring that all relevant documentation needs are met.

2. **Dynamic Document Ingestion:** The tool will enable dynamic and ad-hoc ingestion of existing documents, allowing for seamless integration and utilization of legacy information. This feature will enhance the tool's flexibility and adaptability to existing documentation.

3. **Document Review Assistance:** A document review helper tool will be introduced to assist Quality Managers in evaluating documents against industry standards. This tool will provide improvement suggestions, ensuring the documentation meets the necessary quality benchmarks.

4. **AI-Driven Risk Assessment:** CAIDOC will incorporate AI capabilities to assess risks associated with software failures. This enhancement will offer proactive strategies for risk mitigation, helping teams identify and address potential issues before they escalate.

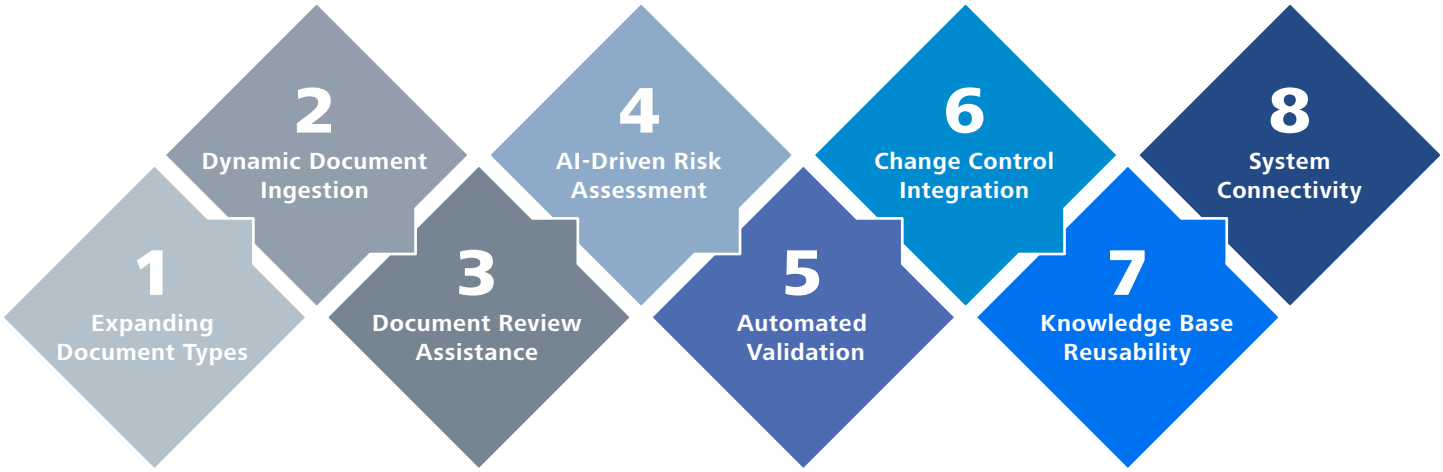
5. **Automated Validation:** AI tools will be implemented to automate validation processes, ensuring that the documentation aligns with project requirements and regulatory standards. This automation will reduce manual effort and increase accuracy in compliance.

6. **Change Control Integration:** A robust AI-driven change control process will be developed to track document and requirement modifications. This integration will enhance communication among team members and reduce errors related to documentation changes.

7. **Knowledge Base Reusability:** Future versions of CAIDOC will focus on creating a centralized knowledge base to facilitate the reuse of documentation across projects. This feature will promote efficiency and consistency in documentation practices.

8. **System Connectivity:** CAIDOC will integrate with external systems that hold requirements, tests, and user stories. This connectivity will provide a comprehensive view of project developments, ensuring that all relevant information is accessible and interconnected.

These planned enhancements will significantly improve CAIDOC's functionality, making it a more powerful tool for managing documentation throughout the Software Development Lifecycle.



3.3 Use Case 3: Agentic Workflow for Test Automation

Context

Comprehensive testing of medical software is key to ensure patient safety and compliance with regulatory requirements. An integral component of modern software testing strategies is unit testing, which involves small, quick-running tests for individual units of software, such as single functions or classes. Unit testing verifies code functionality and is well established as a best practice to improve overall software quality and prevent critical bugs in production. For the development of medical software, unit testing is mandatory to comply with regulatory and safety standards. However, writing and maintaining unit tests, requires significant effort and can lead to high cost. Industry experience, as well as insights from our own projects gathered through expert interviews, indicate that developers may spend up to 30% (about 12% on average) of their total engineering time on unit testing. The required effort can be even greater for safety-critical software, where code test coverage of more than 98% is often mandatory to comply with regulatory standards, such as IEC 62304. Test coverage measures how much of the codebase is exercised by automated tests, typically quantified through unit tests that verify individual functions or components. It is clear, that reducing the time needed for unit testing, without compromising software quality, safety and regulatory compliance, can significantly accelerate development cycles and reduce costs.

A promising approach for achieving this goal is to improve unit test generation with generative AI. AI enabled coding assistants, such as GitHub Copilot, have been shown to significantly reduce the time needed for unit testing by generating tests from source code input and user prompts. However, despite the assistance provided by coding assistants, considerable manual effort remains necessary. This includes tasks such as crafting precise prompts, executing and debugging test cases, and ensuring that the generated tests are of high quality and effectively enhance test coverage. Consequently, a fully automated unit test generation approach could further minimize these efforts and yield substantial additional cost savings – Note: Our initial project was conducted in late 2024 and early 2025. At the time of publishing this report, GenAI-based agentic coding assistants are already capable of effectively generating unit tests underscoring how quickly internal initiatives can become obsolete.

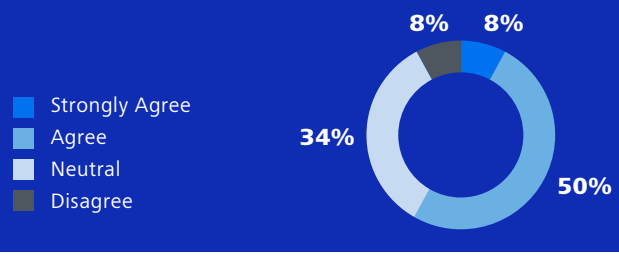
arxiv.org, dx.doi.org, zeiss.com

Developers in exemplary projects spend an average of 12% of their project time writing unit tests to maintain product quality and regulatory compliance. With 58%, a majority of developers either strongly agree or agree that a tool to support unit test generation would alleviate this burden, allowing them to focus on more value-creating tasks, such as developing new features (source: Internal analysis).

How much effort a developer spends on implementing unit tests:
In % of project work

12% on average

A unit test generation tool would free up time to work on new features



As part of our strategy to enhance the software development life cycle through the application of generative AI, we have developed an automated agentic workflow leveraging Large Language Models (LLMs) to generate unit tests directly from source code after having validated the potential in the same way as for the RAG system MVP. This solution augments code coverage by analyzing existing test cases, the code under test, and coverage reports to generate additional test cases that verify previously uncovered execution paths.

The primary goals of our approach include significantly reducing the time and resources required for the manual creation and maintenance of unit tests, which in turn lowers costs and allows developers to focus on core development tasks. By improving test coverage, we aim to enhance software quality by identifying bugs early in the development process, ultimately leading to more robust applications. Furthermore, automating this often unpopular task is expected to boost developer satisfaction, as it reduces the burden of repetitive testing activities.

However, several challenges must be addressed to realize these goals effectively. First, the quality of the generated test cases must be sufficiently high to ensure that the effort required for revision of the generated code does not exceed that of manual implementation. Additionally, the solution must support a variety of technology stacks, including different programming languages, testing tools, and continuous integration (CI) tools. It is also essential for the solution to adapt to company-specific conditions, such as unique libraries and coding patterns. Finally, the ability to generate meaningful tests for complex software systems presents a significant challenge that must be overcome to ensure the effectiveness of our automated testing approach.

Solution Approach

The proposed solution design for our automated unit test generation tool (TestGPT) is depicted in Figure 4. This workflow seamlessly integrates with existing software development tools within a project that are used for building the source code, execution of test suites, and analysis of code coverage. It is designed to be flexible, allowing integration into continuous integration (CI) environments or local use as part of the developer's workflow. When started, the agentic workflow retrieves the code under test from remote or local source code control, along with any pre-existing test cases. This ensures that the most current version of the code and its associated tests are used in the analysis and generation process. The core of the workflow involves AI agents that perform a detailed analysis of the existing tests and the code under test. These agents plan and generate new test cases through a two-step process:

1. **Initial Test Generation:** Tests are initially generated through source code analysis. This step involves understanding the code's structure and functionality and the structure of existing tests to create relevant new test cases.

2. **Coverage-Driven Test Enhancement:** After the initial test suite is built and executed, a comprehensive analysis of code coverage, including line and branch coverage, is conducted. This analysis identifies missed execution paths, which are then fed back into the LLM. The LLM generates additional test cases aimed at covering these missed branches, thereby maximizing coverage improvement.

Existing test cases remain unmodified throughout this process, preserving their original intent and functionality.

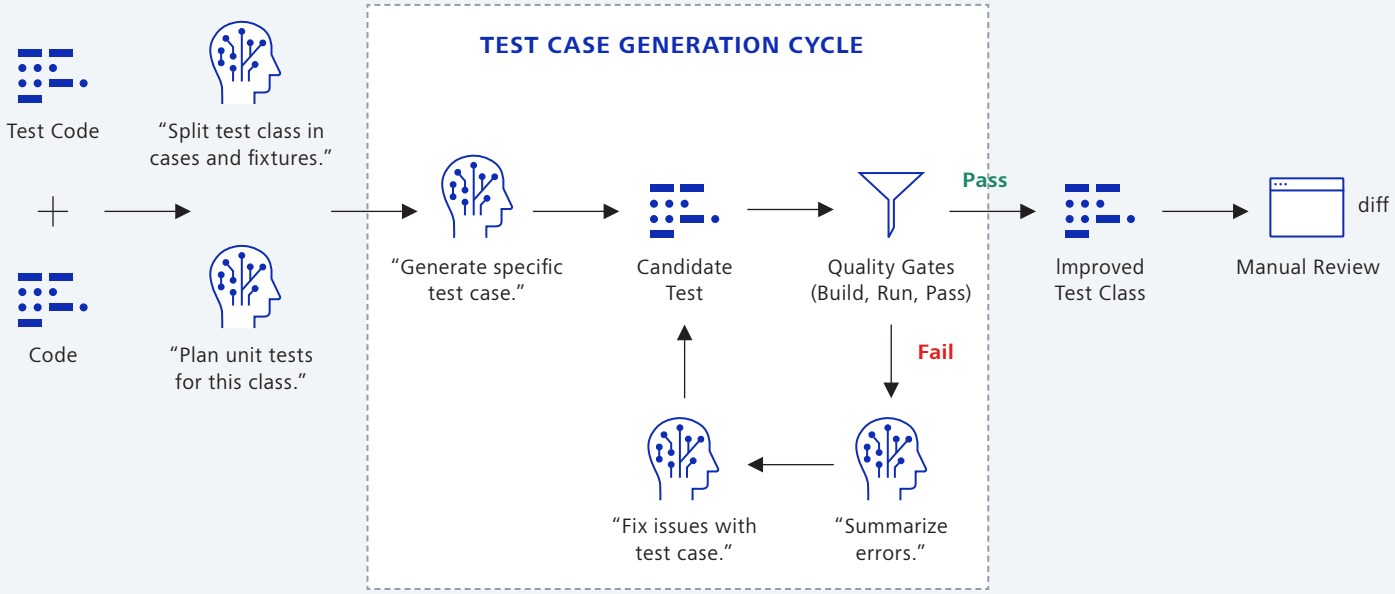


Figure 4: Solution design of TestGPT (simplified)

The LLM used for each agent is configurable, allowing the selection of the most effective and cost-efficient model for each specific task. Figure 5 provides a high-level design overview, but each task is executed by an orchestrated group of small agents, each optimized for a specific function. This micro-agent approach ensures precise and reliable prompting and efficient task execution. LangChain in combination with LangGraph is employed for building and orchestrating these agents, providing a robust framework for managing the complex interactions between agents.

A critical aspect of our workflow is ensuring the high quality of generated tests. Each test is subjected to a series of quality gates, which evaluate the following criteria:

- Successful build of the solution
- Correct execution of the test
- Passing of all tests
- Contribution to increased code coverage

github.com, langchain.com

If a test case fails to meet any of these criteria, agents attempt to debug and rectify the error, after which the test is re-evaluated against the quality gates. Only those test cases that successfully pass all quality gates are incorporated into the enhanced test suite.

In the final step, the enhanced test suite is presented to software developers for a manual code review. This serves as an additional layer of quality assurance, ensuring that only valuable and reliable tests are integrated into the codebase. This process not only maintains the integrity of the test suite but also maximizes the efficiency and effectiveness of the automated testing workflow.

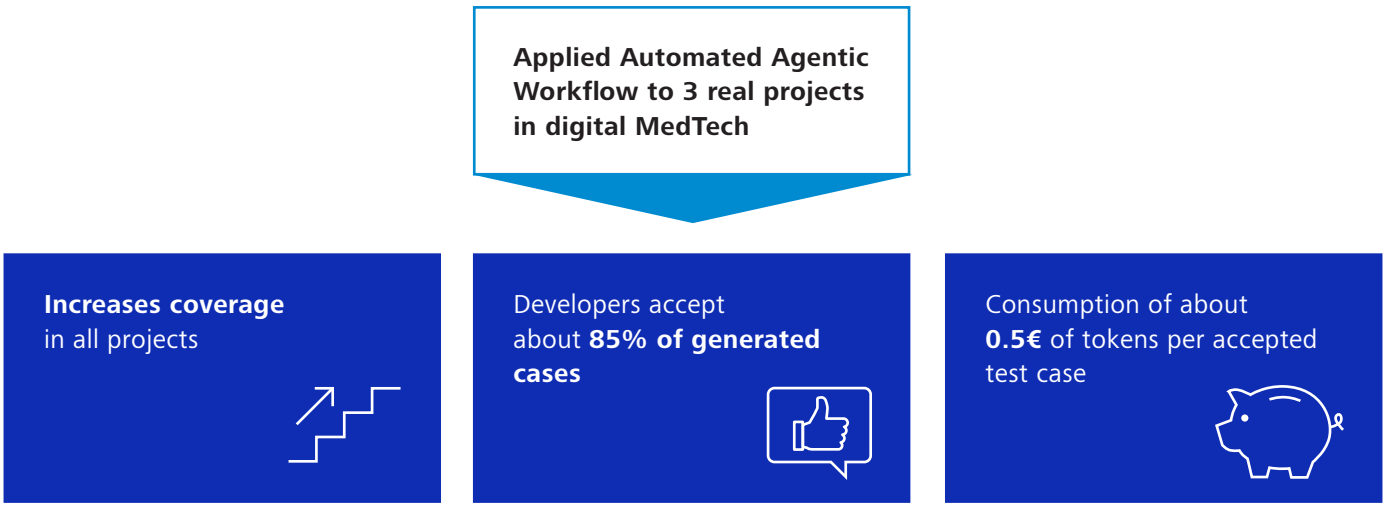


Figure 5 Proof of concept results for the evaluation of TestGPT

Results

The effectiveness of our solution was evaluated using three real-world projects from the digital MedTech sector (Figure 5). Each project employed a different technology stack, providing a comprehensive assessment of the solution's versatility and performance. The projects included:

1. A service and manufacturing tool for medical devices developed in Python.
2. A custom test automation framework for testing medical device firmware, utilizing a C++ stack.
3. A cloud infrastructure solution built with a TypeScript stack.

The results show a substantial increase in code coverage across all three projects (see Table 1). In some cases, coverage values of 100% were achieved for specific classes. It is important to note that, due to the medical context of the projects, the initial test coverage was already relatively high. Despite this, the automated test generation solution was able to significantly increase unit test coverage. This is particularly noteworthy because, as test coverage increases, the effort required to manually write additional test cases typically grows significantly. If the automated test generation solution is used from the beginning of a project where coverage is low, coverage rates can be increased more quickly and efficiently.

	Project 1	Project 2	Project 3
Initial Test Coverage	56%	57%	80%
Achieved Test Coverage	75%	77%	90%

Table 1 Coverage increases through automated unit test generation

However, it is important to ensure that the generated test cases not only inflate coverage statistics but deliver real value for the project. To validate this, the test cases were reviewed by senior developers involved in the respective projects. The acceptance rate of these test cases was 85% across all three projects, indicating a high level of satisfaction with the quality and relevance of the tests produced by our solution. This demonstrates that the generated tests not only improve coverage metrics but actively contribute to the quality of the software.

In terms of cost, the token consumption per accepted test case was only €0.5. This cost efficiency is a critical factor for the scalability and practicality of the solution in real-world applications.

We also compared our approach to a similar solution (Qodo-Cover from Qodo). Both solutions showed comparable performance in terms of coverage increase, with our solution having a slight advantage. However, our solution consumed approximately 40% fewer input tokens, which can lead to significant cost savings when scaled. This efficiency gain is primarily attributed to the more effective micro-agent approach employed in our workflow.

github.com

The evaluation results demonstrate the great potential of GenAI-based solutions to improve code coverage and software quality while reducing costs. The combination of high acceptance rates, substantial coverage improvements, and cost efficiency demonstrates the value of the automated agentic workflow for unit test generation in diverse software development environments.

Lessons Learned

One significant technical hurdle was the varying requirements for test case generation based on the technology stack and project specifics. Each programming language and project has its own semantics and technical details, including how test cases are set up and executed. To address these challenges, we leveraged the analysis results from existing test cases and code. However, an additional project-specific integration layer was necessary on top of the core logic to ensure efficient adaptability to new projects.

Another challenge was ensuring seamless integration into developers' workflows. It was crucial that the automated testing process enhanced productivity without causing disruptions in daily work to facilitate developer acceptance. This was achieved through easy configurability for various tool stacks. In digital MedTech projects, regulatory considerations are critical. A fully automated workflow without human oversight poses significant risks. To mitigate this, we ensured that developers manually review all generated test cases, maintaining the necessary quality and compliance standards, by having a human in the loop.

The implementation highlighted the importance of diverse skill sets within the team. Expertise in software development, software testing, and generative AI technologies is essential. Additionally, generative AI engineers are crucial for optimizing the AI components within the workflow. Furthermore, involving requirements engineers to gather insights from software developers regarding the integration of the tool into their daily work is vital to ensure that the solution meets user needs and enhances productivity effectively.

Next Steps

To further improve our automated agentic workflow for unit test generation, several next steps have been identified. First, we aim to evaluate the solution across additional projects that utilize different technology stacks and application areas. This broader evaluation will help us understand the adaptability and effectiveness of our approach in diverse contexts.

Improving integration into the developer workflow is also a priority, with a focus on implementing features such as IDE integration. This enhancement will streamline the testing process and make it more accessible for developers. Additionally, we plan to improve the solution by incorporating extra quality gates. This could include an agent-based self-assessment of the generated test cases and the implementation of mutation testing to further ensure the robustness of the tests.

Refining existing agents is another critical step, as it will enhance their performance and efficiency in generating high-quality test cases. Moreover, we intend to enrich the solution with comprehensive knowledge of the code base through a Retrieval-Augmented Generation (RAG) approach, combined with a knowledge graph that encompasses the entire code base and project documentation. This enhancement would also facilitate the generation of new test cases from scratch.

Furthermore, we aim to extend the solution to update existing test cases in response to code changes, thereby reducing flakiness and ensuring that the tests remain relevant. Lastly, we plan to expand the scope of our solution to include other testing levels, such as integration testing, which will provide a more comprehensive testing framework and improve overall software quality and software development efficiency.

4. General Implementation Approach

4.1 Data Preparation

Before implementation and deployment, successful generative AI initiatives in healthcare and life sciences require meticulous preparation of proprietary company-internal data. For **text-based applications** leveraging Retrieval-Augmented Generation (RAG), this involves collecting, cleaning, and structuring internal documentation – e.g., technical manuals, SOPs, QMS records, and knowledge base articles – into accessible, high-quality corpora. Data must be segmented, enriched with metadata, and transformed into embeddings using vectorization techniques suitable for semantic search. For **vision-based applications** utilizing fine-tuning of foundation models on domain-specific imaging data (e.g., radiology, pathology, lab assay scans), organizations must anonymize and normalize image data, apply consistent labeling protocols, and ensure ground truth accuracy through expert validation.

nvidia.com, medium.com, softwareone.com

4.2 Implementation & Deployment

Implementing generative AI in healthcare requires a robust technical architecture that supports intensive model training, secure and scalable deployment, and long-term maintainability. Cloud-based solutions are increasingly favored due to their flexibility and scalability, offering access to high-performance computing resources such as GPU clusters, vector databases, and managed AI services that are essential for training and fine-tuning deep learning models on clinical, imaging, or laboratory data. This setup accelerates experimentation, enables dynamic updates, and allows on-demand scaling, making it particularly suited for innovation-driven healthcare initiatives.

On-premises deployments, however, remain crucial for organizations with strict data governance and security requirements. These deployments offer full control over infrastructure and data, facilitating compliance with privacy regulations such as HIPAA and GDPR. In practice, a hybrid architecture where cloud services handle compute-intensive tasks and sensitive data is stored and processed locally is often the most practical and compliant solution.

To support this architecture, key infrastructure components include:

- High-throughput data storage (e.g., Azure Blob Storage, AWS S3, or local NAS)
- Scalable compute (e.g., Azure Machine Learning, AWS SageMaker, or on-prem GPU nodes)
- Vector databases for semantic search (e.g., Pinecone, Weaviate, FAISS)
- Secure API gateways and role-based access control for clinical and research environments

Architectural Approaches for GenAI-Based Systems

A common architectural pattern emerging in GenAI healthcare deployments is the **multi-agent system**. In this approach, specialized AI agents (e.g., for document retrieval, summarization, risk classification, or code generation) operate independently but coordinate to complete complex tasks such as QMS documentation, regulatory submission preparation, or clinical trial reporting. They use an LLM inference service as the backend engine to process reasoning tasks and generate responses for user requests. This modularity supports maintainability, parallelism, and scalability across diverse workflows (Figure 6). The user can securely interact with the multi-agent system via a portal connected to an API gateway.

The agents can obtain specialized data, e.g. from a vector database in the data layer, and are often orchestrated via an agent orchestrator using frameworks like:

- **LangChain** – For building LLM-powered pipelines and connecting tools such as retrievers, memory, agents, and APIs
- **LangGraph** – For defining graph-based workflows that allow conditional logic, retries, and multi-agent interaction patterns
- **Semantic Kernel (Microsoft)** – For embedding LLMs into existing .NET-based healthcare systems
- **Haystack** – For RAG implementations focused on search and question answering over private corpora

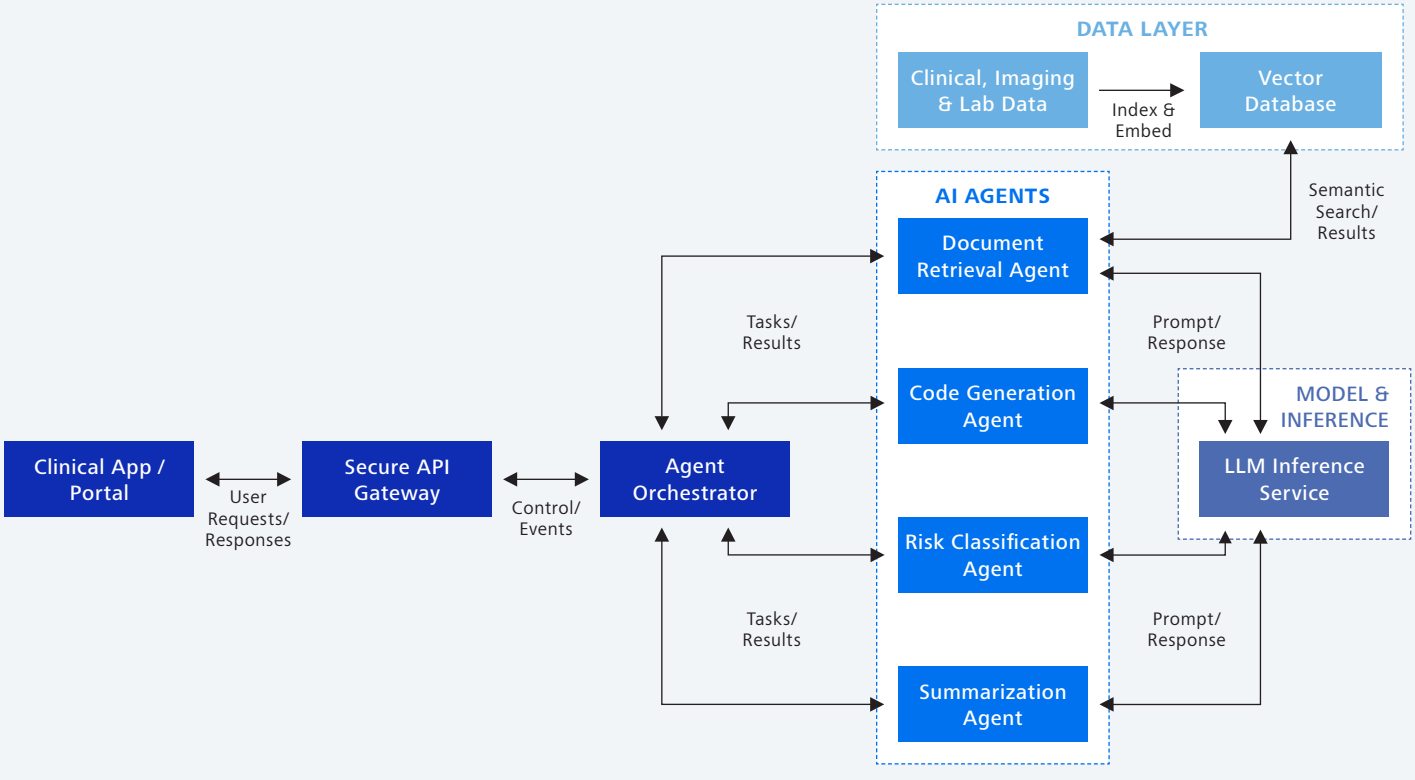


Figure 6: Solution architecture for an exemplary multi-agent system for a clinical application

Cloud Platforms & Deployment Considerations

Cloud ecosystems such as Azure, AWS, and Google Cloud offer native support for deploying GenAI-based solutions:

- **Azure AI Foundry** with integrated OpenAI models and Cognitive Search
- **AWS Bedrock** or **SageMaker JumpStart** for scalable model hosting and data privacy management
- **Google Cloud Vertex AI** for multimodal model integration and healthcare compliance features

For deployment, containerized environments (e.g., Docker, Kubernetes, e.g. Azure AKS, AWS EKS) are essential to ensure scalability, monitoring, and reliability. These platforms allow services such as inference APIs, retrievers, and orchestration agents to scale independently based on usage demands, which is critical in healthcare contexts where performance and uptime are non-negotiable.

To ensure safe and reliable operation, additional best practices include:

- Automated monitoring and logging of model inputs/outputs (for auditability and safety)
- Rate-limiting and approval steps for high-risk tasks (e.g., clinical recommendations)
- Scheduled retraining pipelines to keep the models aligned with evolving knowledge bases and clinical guidelines

amazon.com, google.com, langchain.com, medium.com, microsoft.com, techtarget.com, xenonstack.com

4.3 Maintenance

Maintaining generative AI systems requires continuous monitoring, evaluation, and updates to ensure reliability, compliance, and performance over time. This involves implementing robust **DevOps and GenAI Ops** practices to manage the broader application infrastructure – including APIs, vector databases, orchestration logic, and integration layers – as well as **MLOps or LLMOps** practices to govern the lifecycle of the underlying models, when not fully managed by third-party providers.

Key tasks at the model level (MLOps/LLMOps) include tracking model drift, retraining on updated datasets, validating outputs for accuracy, fairness, and bias, and ensuring reproducibility and governance of model artifacts. In parallel, GenAI-specific operations focus on maintaining prompt templates, tuning retrieval pipelines, managing grounding data sources, and controlling the behavior of large language models – especially in Retrieval-Augmented Generation (RAG) or multi-agent systems.

At the system level, DevOps and GenAI Ops responsibilities include ensuring infrastructure scalability, monitoring uptime, securing APIs, logging user interactions, and managing deployment pipelines across cloud or hybrid environments.

Regular audits and performance evaluations are also essential to meet regulatory standards (e.g., EU AI Act, HIPAA, GDPR) and institutional IT requirements.

Effective maintenance is not a one-time task but an ongoing, multidisciplinary effort that ensures clinical relevance, ethical alignment, and operational stability of GenAI-powered solutions in healthcare and life sciences.

devops.com, medium.com, ml-ops.org, wandb.ai

5. Special Considerations in Healthcare & Life Science

5.1 Data Requirements & Management

Effective implementation of generative AI in healthcare and life science hinges on robust data management strategies that ensure both high-quality inputs and rigorous privacy safeguards. Organizations must establish comprehensive pipelines for data collection, cleaning, and annotation to transform raw clinical, imaging, and genomic data into reliable training datasets. Advanced preprocessing techniques remove noise and standardize formats, while precise annotation enriches these datasets with meaningful labels critical for model performance. Equally important are privacy-preserving methods – such as de-identification and synthetic data generation – that enable compliance with regulations like HIPAA and GDPR while expanding the dataset without compromising sensitive patient information. By integrating these strategies, healthcare providers can harness generative AI to enhance diagnostic tools and accelerate R&D cycles, ultimately fostering innovation and improving patient outcomes.

persistent.com, xenonstack.com

5.2 Regulatory & Compliance Issues

Regulatory and compliance issues form a critical pillar for the successful integration of generative AI into healthcare. Organizations must navigate a multifaceted regulatory landscape, including the U.S. FDA's oversight of Software as a Medical Device (SaMD), HIPAA's stringent patient data protection requirements, and the comprehensive privacy mandates of the EU's GDPR. Additionally, the emerging EU AI Act introduces a risk-based framework specifically regulating AI systems, classifying many healthcare applications as high-risk. This regulation imposes strict requirements on transparency, risk management, data governance, and post-market monitoring making compliance a strategic imperative for any AI deployment in the European market.

Equally essential is the rigorous validation and verification of AI models ensuring that performance metrics such as accuracy, sensitivity, and specificity are robustly tested across diverse datasets and clinical environments to safeguard against bias and overfitting. Building trust and transparency in AI-driven decisions further demands that these systems incorporate explainability and auditability, enabling clinicians, regulators, and patients to understand the rationale behind algorithmic outputs. Initiatives such as standardized reporting frameworks and continuous monitoring practices are critical to maintain ethical oversight and ensurethat AI innovations not only advance healthcare outcomes but also adhere to the highest standards of safety, accountability, and legal compliance.

binariks.com, ncbi.nlm.nih.gov, medium.com

5.3 Ethical & Bias Concerns

Generative AI in healthcare holds the potential to revolutionize patient care, but its adoption must be tempered by vigilant ethical oversight, particularly regarding bias in training data. Traditional medical datasets may underrepresent certain populations, leading AI systems to reinforce existing disparities. For example, a model trained predominantly on data from white patients may perform poorly when applied to minority populations, thereby risking misdiagnoses or unequal care outcomes. Strategies for fairness and accountability include diversifying datasets, employing bias-detection algorithms, and integrating human-in-the-loop systems that ensure clinicians can validate AI outputs. By instituting regular audits, transparent reporting standards, and interdisciplinary oversight, healthcare organizations can mitigate bias and enhance the trustworthiness of AI applications.

pmc.ncbi.nlm.nih.gov, pmc.ncbi.nlm.nih.gov, bdo.com

6. Best Practices & Guidelines

Successful implementation of AI in regulated software development relies on a multidisciplinary team that combines deep technical expertise with specialized medical and regulatory knowledge. Data scientists analyze and prepare complex datasets, while medical domain experts ensure that the AI's outputs align with clinical realities and patient needs. Software and AI engineers develop, deploy, and maintain the systems, and regulatory specialists navigate the intricate landscape of compliance and quality standards such as HIPAA and FDA regulations. This diversity in expertise not only ensures technical robustness but also builds trust among stakeholders by safeguarding patient safety and data privacy. When it comes to development methodologies, AI-driven projects in healthcare benefit greatly from an agile approach rather than traditional waterfall methods. Agile practices facilitate iterative development, allowing teams to quickly adapt to feedback from clinical testing and regulatory reviews. Continuous integration and continuous delivery (CI/CD) pipelines further enhance this process by enabling rapid, reliable updates to AI-based systems. This ensures that new versions are rigorously tested for performance and robustness before deployment, reducing downtime and improving overall system quality. Quality assurance and testing are critical to the success of AI systems in healthcare, where even minor errors can have serious consequences. Rigorous testing strategies must assess not only the robustness and performance of AI systems under various conditions – including edge cases – but also their seamless integration with existing clinical workflows. Integration testing ensures that AI tools work harmoniously with other healthcare IT systems, maintaining data integrity and operational efficiency. This comprehensive approach to testing safeguards against unexpected system behaviors builds confidence in the AI system's reliability. Finally, effective change management and user adoption are essential for integrating AI into clinical practice. Training programs tailored for clinicians and administrative staff help them understand the capabilities and limitations of AI tools, enabling them to use these systems confidently and effectively. Clear communication strategies that articulate both the benefits and constraints of AI-driven solutions are vital to manage expectations and ensure sustained user engagement. This focus on education and transparent communication fosters a collaborative environment where technology enhances patient care without overwhelming end users.

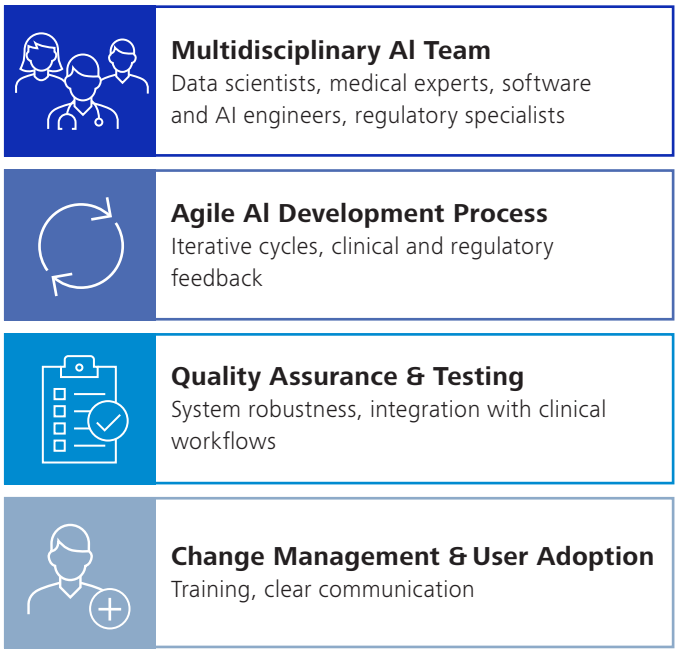


Figure 7 Aspects for AI integration into regulated software development

7. Summary

7.1 Expected Impact

Generative AI is revolutionizing software development by significantly boosting efficiency and reducing engineering cycle times. In general software projects, developers can now document code 45–50% faster, write new code 35–50% faster, and refactor code 30–40% faster – yielding potential savings of 10–15% of total engineering time, which can rise to over 30% with strategic integration.

Key recommendations include structured adoption through focused training, upskilling, and robust governance, alongside workflow optimization by automating routine coding tasks. In healthcare and life science, these benefits are even more transformative, where GenAI automates documentation, diagnostics, and test case creation, supporting biomedical research and predictive modeling with time savings of up to 55% and cost efficiencies of 6–12% of revenue over a few years.

deloitte.com, bcg.com, bcg.com, mckinsey.com, bcg.com, bain.com

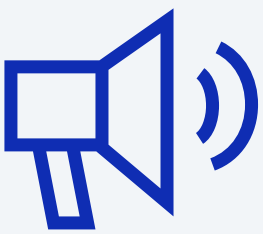
7.2 Quantitative and Qualitative Benefit

Quantitative benefits from leveraging generative AI in regulated software development are multifaceted. For instance, organizations have reported significant reductions in development cycle time – often up to 40–50% – by automating repetitive tasks and streamlining iterative prototyping. These gains translate into tangible cost savings and improved productivity as engineering teams focus on higher-level innovation rather than routine coding or data cleaning. Furthermore, enhanced diagnostic accuracy and workflow efficiencies have been observed, with AI tools delivering up to 20% improvement in key performance metrics by reducing errors in image analysis and clinical decision support processes.

Alongside these quantitative metrics, qualitative benefits further underscore the transformative impact of AI in healthcare & life science. Improved patient outcomes have been achieved through more personalized treatment planning and early detection of diseases, which in turn help reduce hospital readmissions and overall morbidity. Clinician satisfaction is enhanced as AI alleviates administrative burdens, allowing physicians to devote more time to direct patient care and complex decision-making. Additionally, by integrating robust compliance frameworks into the development process, organizations can streamline regulatory adherence – ensuring that AI systems meet stringent standards thus building trust among both regulators and end users.

8. Conclusion

Generative AI is proving to be a transformative force in regulated software development, delivering both quantitative and qualitative benefits that directly enhance patient care and organizational performance. Our analysis shows that GenAI can dramatically reduce development cycle times and lower costs by automating routine tasks. These efficiency gains are complemented by substantial improvements in patient outcomes and clinician satisfaction – allowing healthcare providers to focus on delivering empathetic, personalized care. In today's competitive environment, the strategic value of integrating GenAI solutions extends beyond mere cost savings; it also fortifies regulatory compliance and fosters innovation, giving organizations a significant edge in the rapidly evolving healthcare landscape. We encourage stakeholders to explore these transformative GenAI solutions – whether through pilot programs, collaborative research, or bespoke AI integration strategies – to drive measurable improvements in both operational performance and patient care.



Now is the moment for in healthcare & life science innovators to leverage GenAI – accelerating innovation, enhancing patient outcomes, & achieving strategic advantage. Contact us for further collaboration, detailed discussions, or pilot initiatives to leverage GenAI in your organization.

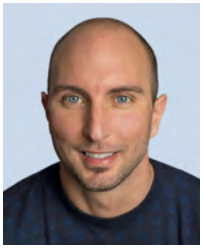
Contact & Lead Author

Dr. Andreas T. Bachmeier
Solutions Enablement Team Lead
ZEISS Digital Innovation
Health & Life Science Solutions
andreas.bachmeier@zeiss.com

Co-Authors

Dirk Asmus
Senior Solution Specialist
ZEISS Digital Innovation
Health & Life Science Solutions

Dmytro Batsenko
Senior Business Development Manager
ZEISS Digital Innovation
Health & Life Science Solutions



David Klusoczki
Senior Requirements Engineer
ZEISS Digital Innovation
Health & Life Science Solutions

Dr. Julian Massing
Senior Solution Specialist
ZEISS Digital Innovation
Health & Life Science Solutions

Maximilian Reichel
Software Developer
ZEISS Digital Innovation
Health & Life Science Solutions

Carl Zeiss Digital Innovation GmbH
Fritz-Foerster-Platz 2
01069 Dresden
Germany

Phone: +49 351 49701 – 500
contact.digitalinnovation.de@zeiss.com
zeiss.de/digital-innovation