accenture

dfki ai Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

Fraunhofer ISST

Dirk A. Molitor, Vlad Larichev, Tobias Guggenberger, Marcel Altendeitering, Daniel Porta, Matthias Ziegler

AI in New Product Development

Connecting Data & Unlocking Knowledge

Table of Content

Executive Summary

Artificial Intelligence (AI) has the potential to fundamentally transform new product development. Applied effectively, it can automate and accelerate engineering processes end to end, from early concept design to product release. Yet this transformative power can only be realized if companies act early and decisively to establish the right technical, organizational, social, and process foundations. Lacking a robust foundation, AI remains confined to pilot successes far from achieving enterprise-wide value.

This white paper presents a scalable framework for AI adoption in engineering, designed to help organizations move beyond fragmented pilot projects toward an enterprise-wide approach. The framework enables leadership to align AI development with strategic business goals while preventing the proliferation of disconnected use cases that dilute value and create complexity.

A central element of the white paper is the classification of AI use cases into two complementary categories.

- **Vertically integrated AI use cases** focus on optimizing specific processes or domains, such as automated design iterations or generation of domain-specific development artifacts.
- **Horizontally integrated AI use cases** connect data, tools, and engineering domains, enabling knowledge sharing, system-level optimization and application across domain and tool boundaries.

While most organizations today concentrate on vertical applications, the greater long-term opportunity lies in horizontal integration. By linking product requirements, architecture, design, simulations, and test artifacts, horizontally integrated AI can unlock cross-domain synergies, accelerate the product development process, and reshape how engineering value is created.

To capture this potential, technology and organization must evolve in tandem. Companies should initiate high priority use cases early to generate learning effects and tangible ROI. At the same time, they must invest in AI-ready infrastructure that ensures interoperability between AI-native platforms and the existing engineering toolchain. Data quality and accessibility become strategic assets, requiring the creation of high-quality data products and the deployment of context modules in the form of knowledge graphs and vector databases to connect data, tools, and processes. Finally, a robust governance framework is essential. Clear guidelines for AI development and lifecycle management will secure alignment with corporate strategy, maintain compliance, and prevent uncontrolled proliferation of use cases.

To ensure the long-term scalability of AI initiatives in new product development, various dimensions must be considered. Unmanaged initiatives will lead to AI fragmentation along existing tools and data silos, preventing the full potential of AI in engineering from being realized.

Companies that act now to establish these technological, organizational, and governance foundations will not only **accelerate their product development cycles** but also create a **sustainable competitive advantage**. Those who delay risk being locked into fragmented solutions and losing pace in an increasingly AI-driven engineering landscape.

Foundations of AI in Engineering

Over the past decades, engineering has undergone a fundamental transformation driven by digitalization. The introduction of computer-aided design (CAD), product data management (PDM) and product lifecycle management (PLM) has evolved into the vision of the digital thread, an interconnected data backbone that links data, information and processes across the entire product lifecycle, from concept to end-of-life. This digital continuity provides engineers with unprecedented visibility and traceability, enabling faster innovation, improved quality, and more efficient decision-making. Many companies still struggle to realize an end-to-end digital thread and attempt to manage the complexity of modern cyber-physical products with rigid, sequential development methodologies and fragmented tool and data architectures, resulting in long development cycles and inefficiencies.

In parallel, AI has matured from experimental research in pattern recognition and machine learning (ML) into a practical tool that enhances nearly every aspect of engineering. In recent years, the emergence of generative AI (GenAI) and Agentic AI has marked a new phase, where AI is no longer only supporting decision-making but actively creating development artifacts across the product development process such as requirement models, product architectures, CAD designs, simulations and test results. This progress is reshaping how engineering teams work, collaborate, and innovate in the future.

To leverage AI in engineering at scale, however, organizations require a structured approach. This white paper aims to provide decision-makers, engineers, and other stakeholders with guidance in the rapidly evolving landscape of AI applications in engineering. In this Section, key concepts are introduced, a scalable framework for the adoption of AI in engineering is presented, and a maturity model based on the automation levels of autonomous driving (SAE 2014) is developed to classify AI applications according to their vertical and horizontal maturity. These foundations are then applied in the subsequent Section "*Use Cases*" to discuss the current state of the art and to analyze 137 research papers. In the "Outlook" Section, we formulate hypotheses on how engineering will evolve under the influence of AI, GenAI, and Agentic AI, and outline the measures companies should take in the short and long term to accelerate their product development processes and eliminate inefficiencies.

Digital Thread

Definition of Digital Thread

A **Digital Thread** is a framework that **seamlessly connects data, models, and processes** across the entire product lifecycle, from ideation and product development to manufacturing and service (Abdel-Aty & Negri 2024). It enables the **continuous flow and accessibility of information** by integrating previously siloed systems and data sources, thereby overcoming data fragmentation across departments and disciplines. By providing a unified, traceable, and context-rich representation of a product's digital history and evolution, the Digital Thread acts as the **future data backbone of engineering, manufacturing, and services**. It supports **real-time decision-making, improves collaboration, and lays the foundation for advanced capabilities** such as GenAI applications and closed-loop engineering feedback.

The Digital Thread represents a core concept for the integrated flow of product data across the entire product lifecycle. By collecting and networking heterogeneous product data across different product representations, a Digital Thread ensures that information and data flow seamlessly along the value chain, creating transparency, traceability, and collaboration

between engineering disciplines. Accordingly, a Digital Thread facilitates the exchange of critical data in product development and enables additional services, the development of new features and the rapid identification of optimization potential (Ghosh et al. 2025). A simplified visualization of these interconnected product representations is shown in Figure 1.

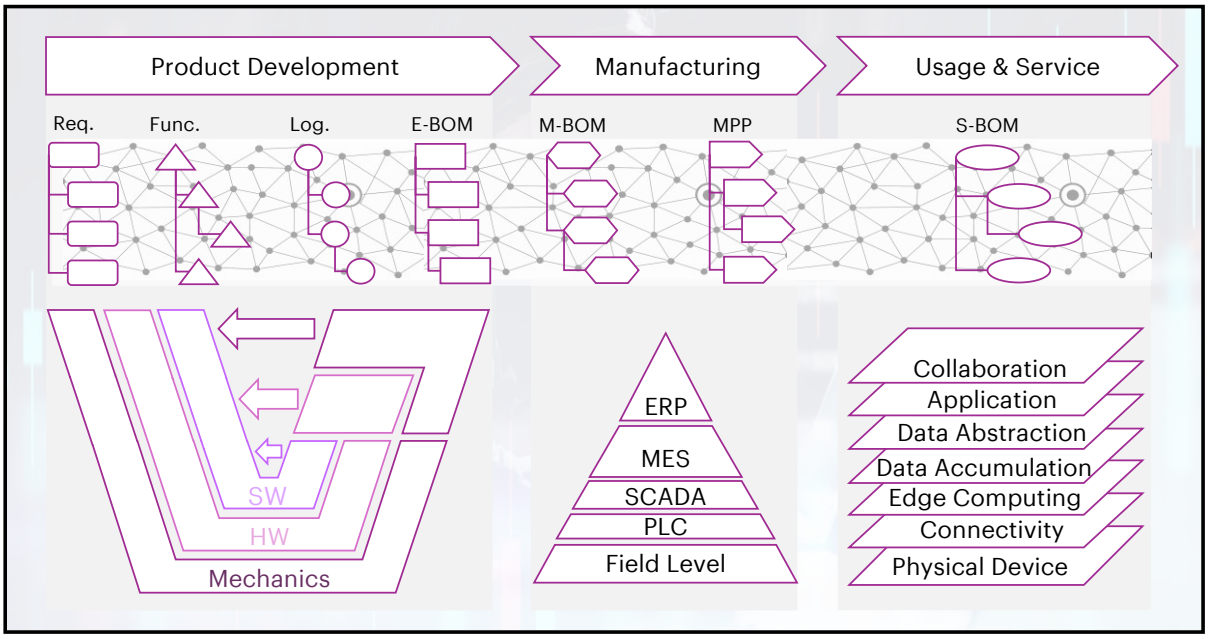


Figure 1: Simplified Visualization of Connecting Product Representations via the Digital Thread

According to Abdel-Aty and Negri (2024), the main characteristics of a digital thread can be summarized as follows:

- **Integration Across Lifecycle Stages**: it links engineering, production, supply chain, and service data, enabling seamless collaboration.
- **Real-Time Data Sharing**: it provides up-to-date information access across departments and disciplines.
- **Improved Decision-Making**: it enhances efficiency, quality, and predictive capabilities across product development and operations.
- **Foundation for Digital Twin**: it supports the creation of a digital twin as a virtual representation of the physical asset.

While closely linked, it is important to distinguish the Digital Thread from the Digital Twin. The Digital Twin focuses on creating a virtual product-centric representation of a physical asset, whereas the Digital Thread focuses on the flow of information and data across the product lifecycle. Companies developing mechatronic and cyber-physical products as well as research institutes recognize the necessity of establishing Digital Threads, which is why most recent publications examine the significance of the Digital Thread (Bianchini et al. 2024, Abdel-Aty and Negri 2024) and show the connections to AI applications (Zhang et al. 2024). Holterman et al. (2024), for example, systematically show how the establishment of Digital Threads contributes to the robustification of supply chains in various industries in the US economy and name AI as a technology that leverages the Digital Thread. Although the concept of the Digital Thread has been researched, solutions for its realization are already offered by software vendors, and the concept theoretically promises significant acceleration of product development processes, companies in industrial practice still struggle with scalable implementation. Data is typically distributed across a fragmented IT and tool landscape, difficult to access, ambiguous, incomplete and weakly connected across different engineering disciplines (Hedberg et al.

2020, Kwon et al. 2020). These circumstances make it difficult to establish seamless workflows across tool and engineering discipline boundaries and require a lot of manual work. This is particularly noticeable in change management processes, where cross-disciplinary work on different engineering artifacts often must be carried out iteratively and the identification of the affected configuration elements or impact chains is time-consuming and resource-intensive (Burggräf et al. 2024).

Building on these challenges, AI technologies and its powerful subfields GenAI and Agentic AI are increasingly recognized as enablers for realizing Digital Threads in practice. By addressing issues of data fragmentation, semantic alignment, and workflow automation, AI can unlock the potential of Digital Threads and make them scalable across industrial environments. The following Subsection therefore introduces and distinguishes the terms AI, GenAI, and Agentic AI and situates their role in the product development process.

AI in Engineering

For a long time, the application of AI in the product development process played only a minor role. However, in recent years, breakthroughs in the fields of GenAI and Agentic AI have marked a turning point. These advances have been accompanied by significant expansions in AI capabilities, which in turn have massively broadened the range of possible applications of AI algorithms within engineering contexts.

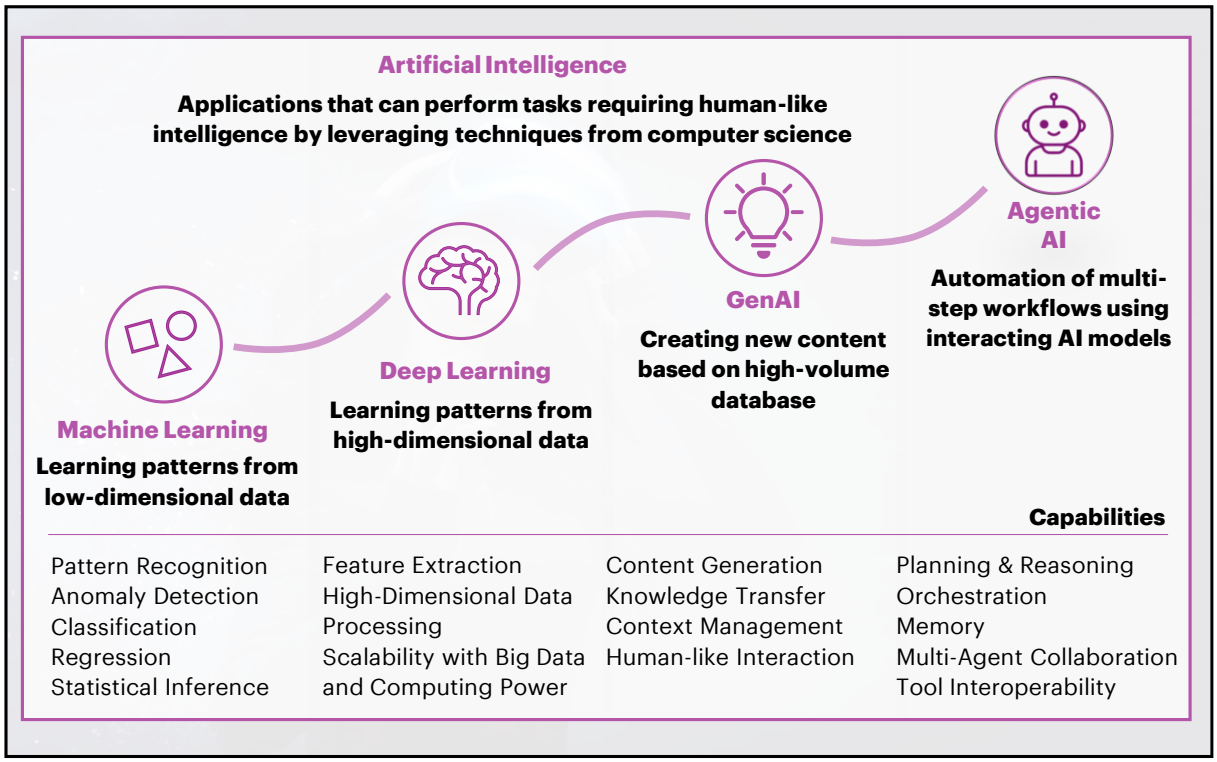


Figure 2: Evolvement of AI Algorithms and corresponding Capabilities

The evolution of AI and the corresponding extensions of its capabilities that emerged with the establishment of distinct AI subfields are illustrated in Figure 2. AI can be understood as an umbrella term for a wide variety of applications in which the automated execution of tasks is enabled by techniques from computer science that originally required human-like intelligence. As research on data-driven models progressed, specific subfields of AI emerged. These

subfields demonstrate different capabilities and impose varying requirements on the underlying datasets.

ML emerged as a paradigm focused on identifying patterns and deriving inferences from structured, often low-dimensional data (Mitchell, 1997). With the advent of Deep Learning (DL), the field shifted toward extracting representations from high-dimensional data, enabling scalability with large and unstructured datasets such as images, speech, and sensor signals (LeCun et al. 2015). The next step in this evolution is Generative AI (GenAI), which leverages large-scale models trained on high-volume databases to create novel content such as text, code, or designs. GenAI thereby expands AI's role from pattern recognition to content generation and contextual interaction, offering new possibilities for knowledge transfer and human-machine collaboration (Cao et al. 2023; Feuerriegel et al. 2024). Most recently, Agentic AI builds on these generative capabilities by orchestrating multi-step workflows through planning, reasoning, memory, and tool interoperability (Wang et al. 2024c). This marks a significant increase in autonomy, where AI systems are no longer limited to generating outputs but can act as agents within complex engineering and product development environments.

Taken together, these stages show not only a rapid technological evolution but also an expansion of potential applications in the product development process: from supporting low-dimensional data analysis (ML), through managing complex engineering data (DL), to assisting in creative design and knowledge-intensive tasks (GenAI), and ultimately enabling partially autonomous management of iterative, cross-disciplinary workflows (Agentic AI).

Before the full potential of the presented AI subfields can be realized in product development processes, certain prerequisites must be established. While companies can already implement high-value AI use cases that deliver a fast Return on Investment (ROI), focusing solely on isolated use cases risks reinforcing the very challenges many organizations already face in relation to data, tools, and processes: fragmentation. To avoid this, it is essential not only to identify and implement high-value use cases but also to create the structural and organizational conditions that allow AI in engineering to scale sustainably.

We argue that these approaches are not mutually exclusive but can complement one another. On the one hand, implementing high-value AI use cases within individual engineering disciplines can lay important groundwork for the Digital Thread. On the other hand, adopting a "Thread-First" perspective *(Accenture Research Report 2021)* ensures that strategic objectives and key capabilities are defined and continuously monitored, guiding the selection and execution of AI initiatives. Such a combined strategy leverages synergies: it enables organizations to capture rapid benefits through GenAI use cases while simultaneously ensuring the long-term scalability of these solutions, ultimately helping to overcome data silos and fragmented IT landscapes.

To ensure a scalable application of AI in the product development process, several dimensions need to be considered and systematically consolidated within a unified framework. For this reason, the following Subsection introduces a framework consisting of five key dimensions, which should be considered in every engineering AI transformation.

Framework for Scalable AI in Engineering

The current starting point for most engineering companies developing complex, mechatronic product is challenging and characterized by

- inadequate data flows / connectivity across lifecycle stages,
- persistent data silos,
- proprietary and non-standardized PMT departments,
- a lack of interoperability between engineering tools, methods, and disciplines,
- redundant, ambiguous and incomplete data,
- missing real-time access to consistent product information,
- the involvement of many expensive tools.

The root cause lies in historically grown engineering tool chains, workflows, and collaboration models. Efforts to optimize individual disciplines have led to a fragmented landscape of tools, data and processes, often lacking a holistic product-level perspective. This situation impedes the realization of end-to-end traceability, consistent data management, and efficient cross-domain collaboration.

Addressing these challenges requires deliberate efforts to establish a connected engineering landscape that builds upon existing solutions. Central questions arise:

1. *How can companies modify their brownfield environments to achieve maximum value with manageable efforts?*
2. *How can engineering be prepared for the future, where AI will play a pivotal role?*
3. *What steps should companies take to rapidly address high-value AI use cases while simultaneously building the foundations for scalable AI strategies in the product development process?*

To answer these questions, we propose a framework for the scalable application of AI in engineering (see Figure 3) that helps organizations to pay attention to key capabilities, which should be considered and monitored during the implementation of AI use cases and serve as the foundation for a Digital Thread & AI adoption.

Our framework focuses on five key dimensions critical to establishing an AI-enabling Digital Thread:

1. **Data Quality**: Increasing data quality consists of ensuring the use of a limited number of consistent data formats for engineering artifacts across the product lifecycle and eliminating redundant, ambiguous and incomplete data. We propose a decentralized data architecture based on the data mesh concept (Dehghani 2022), which bundles domain-specific data into data products and catalogs. Data is made available for the application of AI use cases. The overall goal of the Data Quality dimension is to provide accurate, reliable, and understandable data products from the engineering toolchain that are accessible to data consumers. The aim is not to guarantee data completeness, accuracy, and uniqueness across all data sets, but rather to ensure the necessary level of data quality for particularly important data sets and facilitate reliable AI applications.
2. **Interoperability**: Data is generated, managed, and maintained within the engineering toolchain. To enable both accessibility and modification of this data by AI applications, bidirectional communication between the individual tools and the AI platform is essential. This communication is facilitated through an interoperability layer positioned between the engineering tools (and their data products) and the AI platform. The

interoperability layer ensures seamless data transfers, synchronization as well as agent-based communication, thereby creating the foundation for efficient and scalable integration of AI into engineering processes.

3. **AI Platform**: The AI Platform and its infrastructure ensure the technical realization of AI use cases. The technical infrastructure consists of polyglot data storage solutions, computing capacities and connections to cloud or on-premises systems. The AI applications are then implemented on the AI Platform using microservice architectures and containerized CI/CD pipelines, which are continuously updated and monitored.
4. **Context Management**: Knowledge from the entire product development process should be made accessible and structured at central points. For this purpose, knowledge is aggregated on the AI platform within context modules, which are represented in the form of knowledge graphs (KGs) and vector databases. The objective of these context modules is to capture the relationships between the development artifacts contained in the domain-specific data products and enabling quick access to relevant information. Access to model-based systems engineering (MBSE) tools and (cross-domain) process models that represent product and process modeling at the metamodel level is also an important source of context for providing GenAI and Agentic AI applications with the information required for handling complex engineering tasks.
5. **Federated Governance**: The Federated Governance Team is responsible for the overall management of the AI use case landscape. This includes the definition of standards and interfaces for data transfer, the strategy definition for use case selection and the specification and continuous monitoring of targets and KPIs. Specifying common, cross-domain processes and ensuring interoperability between domains is the central task of federated governance, without interfering too deeply in the domains' areas of responsibility. To ensure secure and compliant use of product data, solutions must provide dynamically managed usage rights while guaranteeing compliance with regulations, safeguarding data ownership, enabling secure third-party interfaces, and meeting cybersecurity requirements.

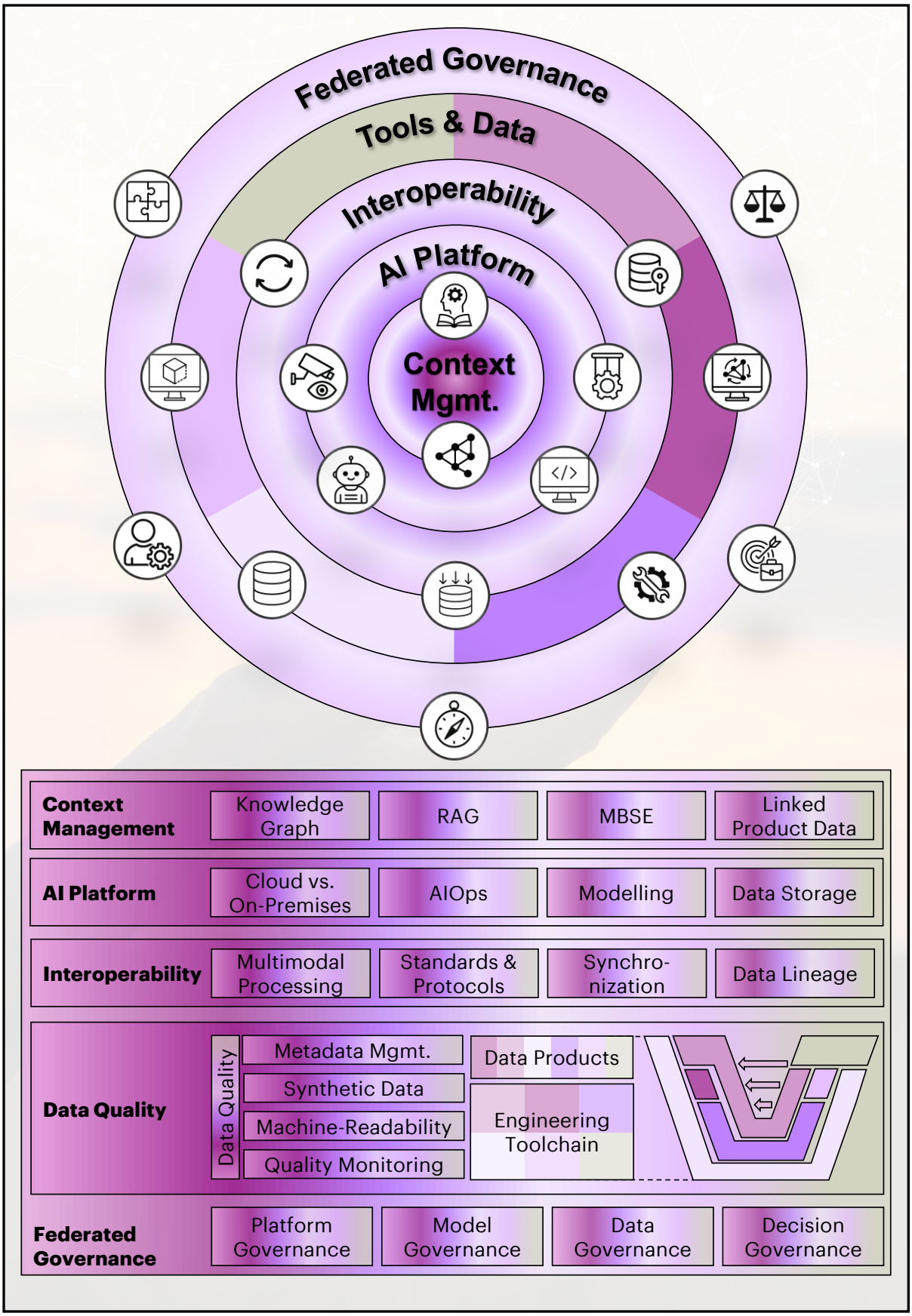


Figure 3: Proposed AI in Engineering Framework with five Key Dimensions

The following Subsections describe the five key dimensions in detail and provide insights into the components that constitute these key dimensions. In addition, experts from the respective domains share perspectives on industrial implementations and report on practical experiences.

01 Data Quality

Ensuring high data quality in engineering tool landscapes is a critical challenge for any enterprise. Engineering activities generate and manage a wide range of artifacts across multiple tools. As products become more complex with increasing E/E and software integration, data heterogeneity increases, and the number of different data formats expands, leading to an exponential rise in data management complexity. Each engineering domain typically works with its own data models, formats, and lifecycles. To effectively manage this complexity, a data mesh approach with a decentralized data architecture (Dehghani 2022) is recommended, which is well-suited for engineering environments (Hooshmand et al. 2022). In such a setup, domain teams act as data product owners, leveraging their deep understanding of domain-specific data and formats. Domain-specific data is bundled into consumable data products and provided to a variety of data consumers across the organization.

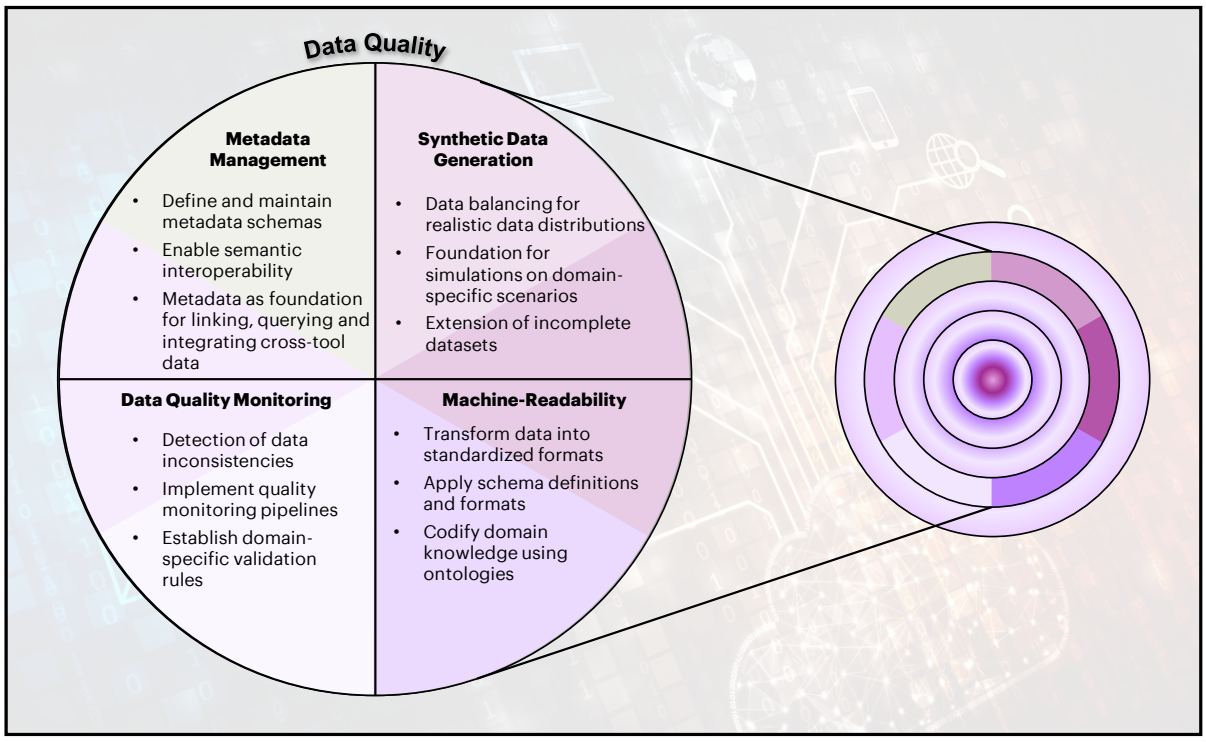


Figure 4: Pillars of Data Quality in Engineering Tool Landscapes

Figure 4 shows the four most important pillars for ensuring high data quality. The pillars are discussed below in the context of product development.

1. **Machine-Readability**: A key objective is to maximize data reuse across the enterprise while minimizing the number of data formats in use. To achieve semantic consistency and machine-readability, particular emphasis must be placed on the codification of engineering artifacts, transforming heterogeneous data into structured, machine-readable formats and embedding them in schema syntaxes (Hooshmand 2022). Engineering data is highly heterogeneous, which is why different domains have their own standards and data formats (see Figure 5). The conversion of unstructured data into defined schemas and the introduction of domain-specific semantic rules is a basic

prerequisite for making data machine-readable and thus preparing it for use by AI algorithms. For example, unstructured data in industrial companies contains a wealth of information that is extremely important for AI applications (Tinnes et al. 2024), AI algorithms can extract valuable information from it (Mahadevkar et al. 2024) and serve as data preprocessors (Zhang et al. 2023). Care must be taken to ensure that the outputs of the algorithms are structured and correspond to the specifications of the desired data formats (Liu et al. 2024b). Especially in engineering, there are many different, complex technical documents and files from which a variety of information can be extracted. Jamieson et al. (2024) provide a comprehensible overview of AI applications for the processing of technical engineering documents. Zhang et al. (2025b) summarize different use cases that derive textual descriptions and annotations from CAD models. A structured analysis conducted in 2022 within the AI Marketplace initiative assessed 23 common data formats and data models from the product development process for their suitability in serving as AI model input and demonstrated that, at this point in time, many of them require transformation processes that result in data loss (AI Marketplace 2022).

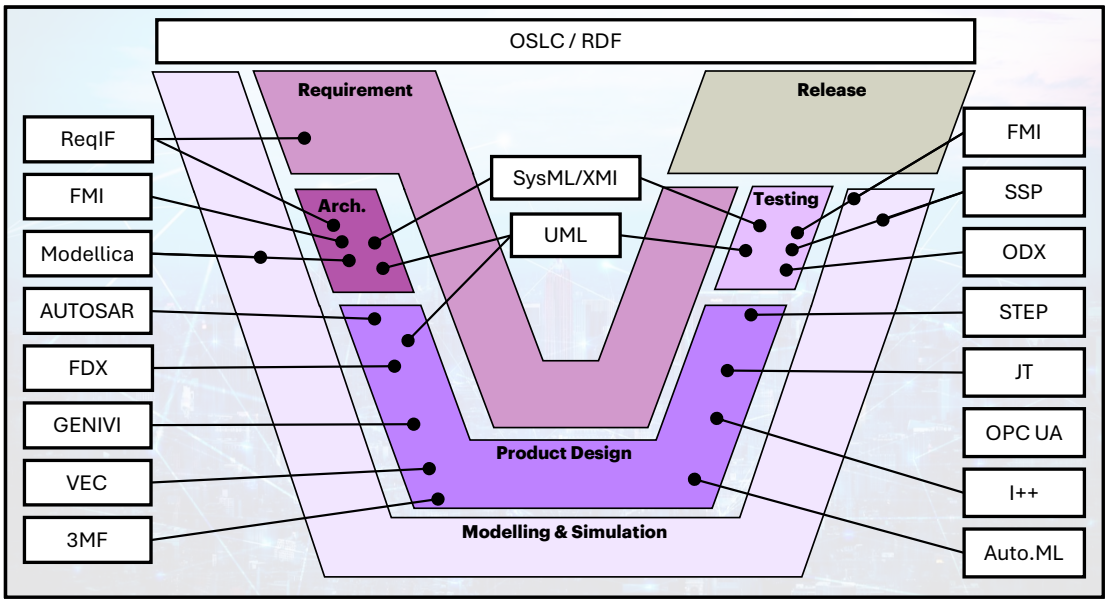


Figure 5: Typical Data Formats, Models and Standards in the Engineering Domain based on prostep ivip (2025)

In the future, the reusability of machine-readable data from past product development cycles and the use of standardized data formats will be one of the most important success factors for the scalable application of AI in product development and should therefore be part of every engineering AI transformation.

2. **Data Quality Monitoring**: To ensure compliance with domain-specific schemas, a range of data quality tools have been developed to continuously monitor, evaluate and improve data quality (Altendeitering & Guggenberger 2024, Altendeitering et al. 2024). Typical cases are the identification of inconsistencies, missing values, redundancies or outdates, where data quality tools either make automated modifications or guided recommendations to the user. AI algorithms can provide support for highly repetitive tasks, such as cleansing data (Narayan et al. 2022) and converting unstructured data into machine-readable formats (Zhu et al. 2024). The creation of monitoring pipelines and the introduction of domain-specific validation rules is recommended to enable the most efficient possible bundling into data products and to improve their quality. Zhou

et al. (2024b) provide a comprehensive review of data quality dimensions and tools for AI applications. It is important to note that while accuracy, completeness, and unambiguity of data are desirable objectives, achieving them in absolute terms is either unattainable or only possible at unjustifiable effort. In the context of an AI engineering transformation, this raises the question of which data are particularly critical, which need to be monitored, and to what extent data quality must be improved to generate reliable AI outputs.

3. **Metadata Management**: Among all data layers presented in Figure 3, metadata plays a critical role and is decisive for algorithms to understand the horizontal and vertical complexity of a mechatronic product (Bode et al. 2024). Metadata provides the semantic context needed for machines and humans alike to understand and process data. It forms the foundation for semantic interoperability, a prerequisite for data sharing and integration activities. Therefore, engineering artefacts should be consistently described by metadata, whereby GenAI can support the creation, standardization, and maintenance of metadata (Yang et al. 2025). Metadata is particularly important in the framework presented in Figure 3, as it is fundamental to two of the key dimensions presented: interoperability and context management.

4. **Synthetic Data Generation**: When only incomplete datasets or those that do not cover the full solution space are available, enriching them with synthetic data is an effective way to improve the performance of AI algorithms. Synthesizing data can also serve as a valuable fallback, especially in domain-specific scenarios where little or no real-world data is accessible, such as early development phases, edge cases, or rare failure modes in engineering systems. In engineering domains, the generation of synthetic data is gaining increasing importance, as real-world data collection is often time-consuming, expensive, or limited by operational constraints. Techniques for synthetic data generation range from simple rule-based simulations to advanced generative models like Generative Adversarial Networks (GANs) or diffusion models. These synthetic datasets can be used to train, validate, and test AI models under a variety of conditions and have proven to enhance AI performance in different domains, such as requirement management (El-Hajjami & Salinesi 2025), MBSE (Muttillo et al. 2024) and scenario testing (Song et al. 2025).

In summary, managing data quality in engineering tool landscapes is fundamental for the success of any AI use case and requires a shift towards decentralized data ownership, as in-depth knowledge of domain-specific data is only available in the domain teams. Recognizing weaknesses in data management and tackling them with the pillars presented is an important first step towards the transformation into data-driven engineering. LLMs have recently proven that they are suitable for a range of tasks to increase data quality (e.g. Naeem et al. 2024) and often the first valuable AI use case is not to optimize (engineering) processes, but to improve data quality (Singh 2023).

Industry Insights into Data Quality Management

The increasing number of data sources, volumes, and formats makes data quality management a complex topic. The processes for identifying, analyzing, and resolving data quality issues are often manual and cumbersome. Moreover, the necessary metadata in the form of data profiles and data quality rules is often not available, which further complicates data quality management. As a result, data quality problems often remain undetected and lead to process disruptions.

To address this problem, we implemented AI and ML technologies at multiple points of the process. At the beginning of the process, we utilized ML algorithms for automated data profiling and generating data quality rules for multiple attributes to identify dependencies. Based on these solutions, we used established GenAI models to generate metadata and transform the identified data quality rules into SQL code.

An important insight of the use case was that AI and ML technologies are very well-suited for identifying complex data quality rules and accuracy problems involving multiple attributes. These are often missed by humans. Additionally, the automated generation of SQL code helps reduce the manual effort required for creating data quality rules.

Marcel Altendeitering
Head of Department
Fraunhofer ISST

Tobias Guggenberger
Group Lead
Fraunhofer ISST

02 Interoperability

In the framework presented in Figure 3, we present multiple levels on which data is managed and made accessible. Data must be transferred between engineering toolchain, its data products and the AI platform and its context modules. The transfer should be as efficient and as close to real time as possible to ensure up-to-date representations. Interoperability refers to the exchange of data between these levels and thus pursues the goal that different product representations, artifacts and metadata are continuously synchronized and thus the same product data with different levels of detail is available at all levels. Access to the latest data is particularly important for AI in engineering, as relationships between domains, disciplines, and development teams evolve dynamically, and the performance of AI algorithms can only be guaranteed if accurate and up-to-date context is provided (Mei et al. 2025). Figure 6 shows the four most important pillars for enabling interoperability, which are further discussed below.

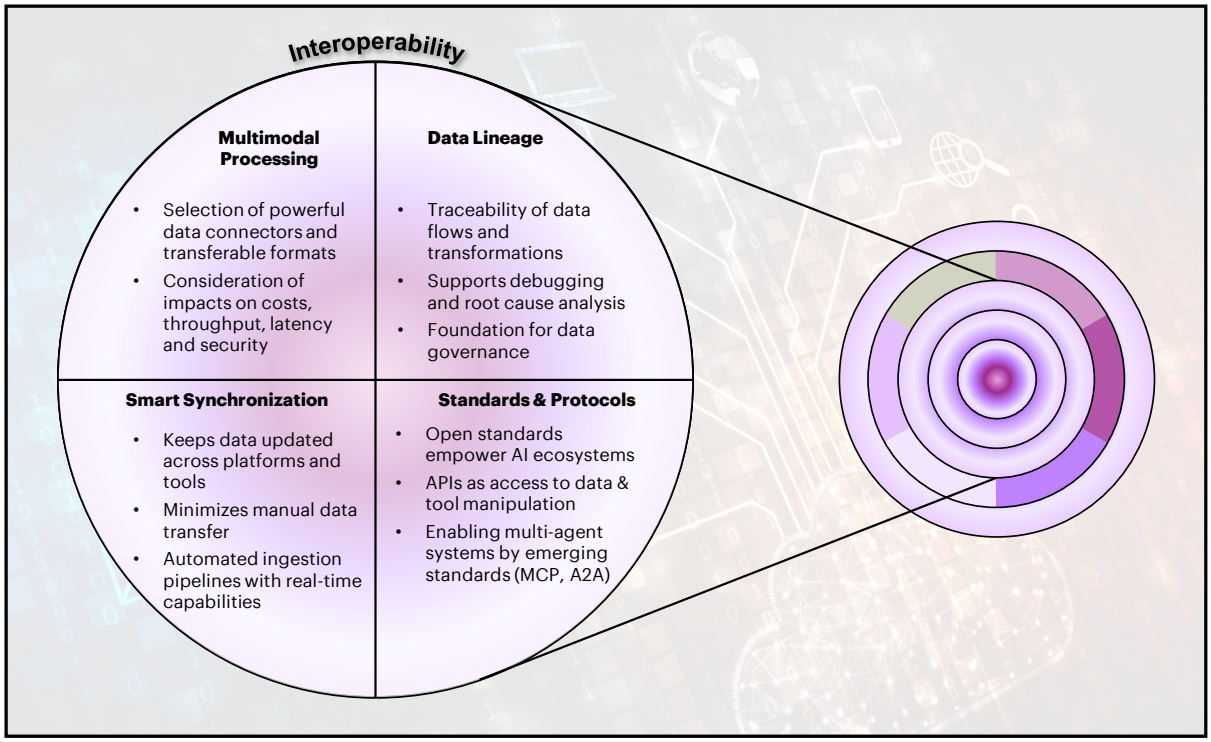


Figure 6: Pillars of Interoperability in Engineering AI Landscapes

1. **Multimodal Processing**
   Engineering data are inherently heterogeneous, appearing in many formats and dimensionalities throughout the product development process (see subsection *Data Quality*). This is why data connectors must be tailored to accommodate these variations and preserve context. Engineering data combines 3D models, simulation results, test signals, and rich metadata. Some arrive as very large files, others as fast, continuous streams. To keep the levels of the framework synchronized, the interoperability layer needs connectors that can move these different data types efficiently and keep their context linked. Research on Digital Thread shows that lifecycle analytics only work when heterogeneous data stays connected across systems, not just copied in pieces (Abdel-Aty et al. 2024). At the same time, studies on big-data transfer in cloud environments show that the choice of the transfer approach strongly affects throughput, latency, cost, and security (Majigi et al. 2025), which is why connectors should support both bulk movement and real-time streaming while preserving metadata for traceability. Especially the transfer of high-dimensional data such as CAD and simulation models is challenging due to large file sizes and heterogeneous formats. Recent studies show that converting these assets into HDF/HDF5 can streamline movement of memory-intensive CAD models (Khan & Rezwana 2021) and simulation datasets (Kunc & Bröcker 2024; Bröcker et al. 2024), which has the potential to improve interoperability between engineering tools and AI platforms in the future.

2. **Data Lineage**: Data lineage, closely linked to the concept of data provenance, refers to the lifecycle and movement of data, enabling the capability to identify the data source (input) and destination (output), including all transformations, processes and intermediate steps, at any point in time and in any system. Especially when data is transferred multiple times between the levels of the framework, it is important for regulatory and validation purposes to comprehend the origin of the data to be able to continuously assess data quality and reliability. Corresponding information can be

stored in metadata, which is why data lineage is closely linked to the quality assurance of data (see key dimension *Data Quality*). In cases where the performance of trained AI models is unexpectedly poor, data lineage approaches help identify errors in the datasets and uncover their root causes (Pahune et al. 2025). Data and AI form a sociotechnical construct in which effective collaboration between domain and AI experts working on different levels in the framework depends on a shared understanding of the data: an understanding made possible through data lineage, which serves as a prerequisite for transparent communication about data origins, context, and transformations (Jarrahi et al. 2023).

3. **Smart Synchronization**: To be able to access the most up-to-date data at all levels, the data must be synchronized without creating copies. Access to current data in real time is a key feature for the engineering of the future, as it allows real-time optimization between levels and even the integration of data outside of engineering (manufacturing, maintenance & service), as emphasized by Ghosh et al. (2025). Data federation enables such access by allowing users to query multiple, heterogeneous data sources through a unified interface, without duplicating or moving the data. This minimises the amount of integration work required, helps maintain up-to-date data, and enables real-time analysis.[1] (Gu et al. 2024) Complementing this, modern data stream processing systems, as highlighted by Fragkoulis et al. (2024), provide the technical backbone for real-time synchronization by enabling stateful, low-latency, and fault-tolerant integration of live data flows across distributed engineering tools and systems. It is important to recognize that valuable data does not just come from engineering tools, but also includes information from production systems, sensors, and IoT devices, offering insights from the manufacturing and service phases and enabling the vision of closed-loop engineering (Durão et al. 2024). Regarding synchronization, users should ask themselves which data is exchanged in which format and at what frequency between the levels to minimize costs and latencies as much as possible.

4. **Interoperability Standards**: Especially dealing with engineering data, a lot of interoperability challenges can be faced, such as different standards and specifications, lack of semantics, lack of communication mechanisms and protocols, high complexity and costs, lack of trust regarding data sharing and security/privacy concerns and scalability (Liepert et al. 2024). Interoperability and integration must not only be considered in terms of streaming data from engineering tools into the AI platform, but also in the reverse direction: insights gained from data-driven analysis at higher system levels must be fed back into the engineering tools and result in (human-supervised) modifications of engineering artifacts. This bidirectional flow is essential to enable data-driven engineering. Many standardized technologies for data exchange are available nowadays. Most widely used are APIs, which are implemented in almost every modern tool or platform. Standardized data exchange between tools and platforms can also be enabled through ETL data pipelines (Foidl et al. 2024) and message queues (Maharjan et al. 2023). Additional emerging technology standards that significantly improve tool accessibility and agentic communication are Anthropic's Model Context Protocol (MCP) (Hou et al. 2025b) and Google's Agent-to-Agent (A2A) protocol (Ray 2025). MCP ensures

[1] Gu et al. (2024) provide a comprehensive survey of data federation systems, analyzing 51 solutions using a structured evaluation framework. Their work identifies key capabilities (e.g., query languages, security features, supported data types) and highlights data federation's value in enabling integration of distributed data without compromising freshness or consistency.

standardized communication between agents and (engineering) tools, while A2A enables standardized communication between multiple agents in multi-agent systems. Figure 7 illustrates a MCP workflow for an engineering use case. The engineer interacts with an MCP Host, which represents an application, and submits a prompt to the MCP Client, which the MCP Host manages. Based on the prompt, the MCP Client establishes a connection to an MCP Server that has access to one or more engineering tools. Through API invocation and the use of pre-defined prompt templates, the server can either retrieve data from the tools or execute tasks within them. During this process, the client and server communicate bidirectionally until the task is completed. The engineer then receives a response from the client regarding the task's completion.

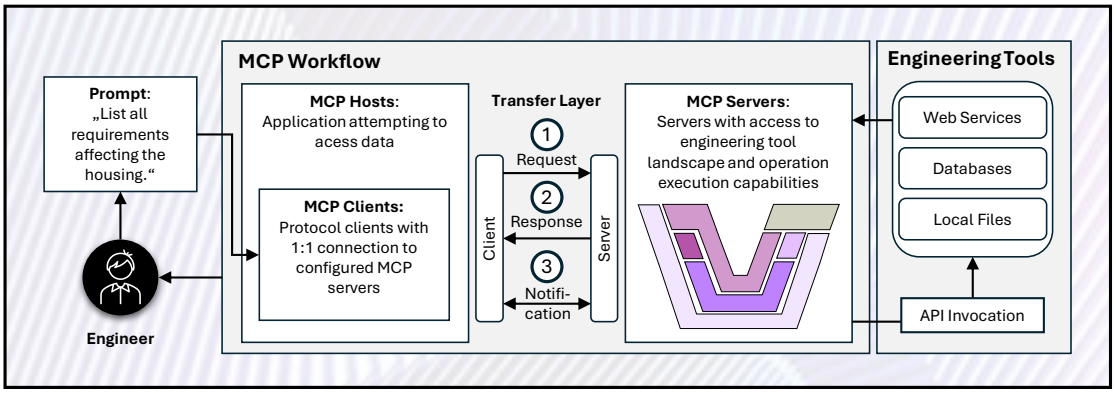


Figure 7: MCP Workflow based on Hou et al. (2025b) adapted to an Engineering Application

In the example provided, the engineer ideally receives a list of all requirements related to the housing, which are managed within the requirement management tool accessible by the MCP Server. In the future, MCP implementations will mean that routine tasks such as data calls or simple modifications to engineering artifacts will no longer have to be performed within the designated tools, but can be carried out using simple prompts on an AI platform.

In summary, interoperability is essential to ensuring a consistent, real-time representation of engineering data across different abstraction levels in the enterprise IT landscape. It enables synchronized product views from detailed engineering artifacts to high-level knowledge graphs and AI-ready datasets. Interoperability requires multimodal data processing capabilities, domain-level ownership, and clear traceability to ensure transparency. Real-time data synchronization and federated access help keep information up to date without duplication. Standards and integration technologies like APIs and MCP or A2A enable smooth data flow between systems, turning AI insights into engineering actions. This lays the groundwork for data-driven engineering and faster, data-driven decisions.

Industry Insights into Data Streaming & Product Data Changes

At a German Automotive Company, bridging the gap between Car Design and Component Manufacturing has always been a challenge. Meeting it means combining data sources for Bills of Materials, Configurations and Supply Chains.

We used the Streaming Service Kafka to connect various source systems to a central cloud-based data hub which does the calculation. Sources communicate a change to their data via message which is put into a topic in the stream. Kafka lets the receiving end choose when to read these messages.
There is a dilemma, though: Product data is massive, reader's interests are very diverse, changes happen frequently, calculating them is costly. Therefore, senders have an incentive to add content to the messages to provide for more receiving parties at the same time. On the other hand, receivers prefer smaller, more relevant topics. Addition of content means more messages. A small amount at first, but the growth was exponential and overwhelmed our receiver soon. What to do?

As we could not limit the topics' sizes and senders were unable to flag the changes, we established an "intelligence kernel" in the receiver. A concentrated, quick to apply version of the main intelligence which is itself constantly updated in case of relevant changes. It reduced the number of irrelevant messages by more than 99%.

Sebastian Mario Jülich
Solution Architect
Accenture

Tamás Biró
Development Lead
Accenture

03 AI Platform

In addition to decentralized data architecture, engineering enterprises require a centralized AI platform where AI use cases can be developed providing required storage and computing capacities. Modern AI algorithms, particularly LLMs and agent-based systems, depend on powerful infrastructures consisting of computing resources, storage, and containerized development environments. Selecting the right vendors, embedding the AI platform into the overarching enterprise architecture (Ettinger 2025), and designing the architecture for large enterprises is a highly complex process (Ismail et al. 2025, Eken et al. 2024), but crucial for AI adoption in engineering.

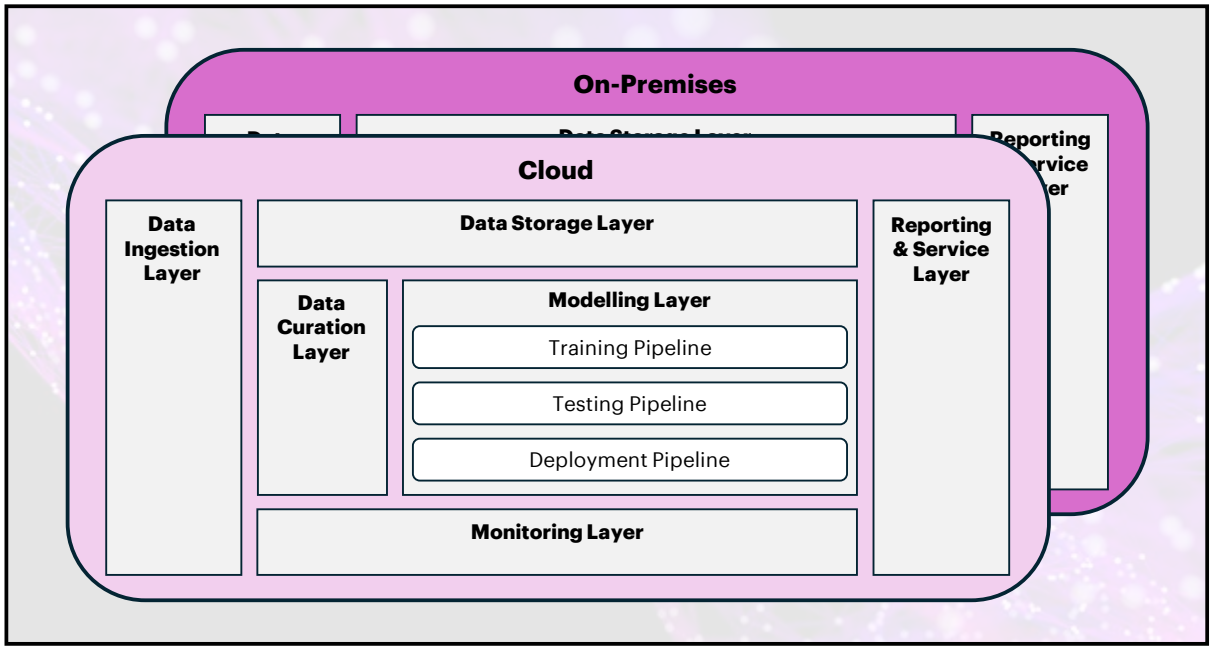


Figure 8: High-Level AI Platform Architecture

Figure 8 shows a high-level architecture of a central AI platform with the most important layers. With the ingestion of data (see key capability *interoperability*), data is fed into the platform and

then processed in the curation layer using preparation, extraction and transformation techniques. The processed data is stored in appropriate formats across suitable databases (polyglot storage) and is readily available for direct input into the models. The modelling layer includes containerized pipelines for training, testing, and deploying the various AI models. In the monitoring layer, the performance of the AI models is continuously assessed, and appropriate actions are taken in case of performance degradation. The reporting & service layer serves as the interface to the user, presenting the results of the model applications through visualizations such as dashboards.

Poor AI platform architecture decisions can have severe consequences and jeopardize the success of the entire engineering AI adoption. Furthermore, cybersecurity concerns must be addressed, ensuring that company data and the associated intellectual property are continuously protected (Admass et al. 2024). This is especially critical in highly regulated industries or when handling data that directly has impacts on the company's competitiveness. In such cases, the question often arises whether sensitive data should be processed in cloud environments or whether on-premises solutions are the better choice (see also key dimension *federated governance*). Based on four pillars (see Figure 9), we outline the key capabilities that should be considered in designing the AI platform and the potential implications of various design decisions.

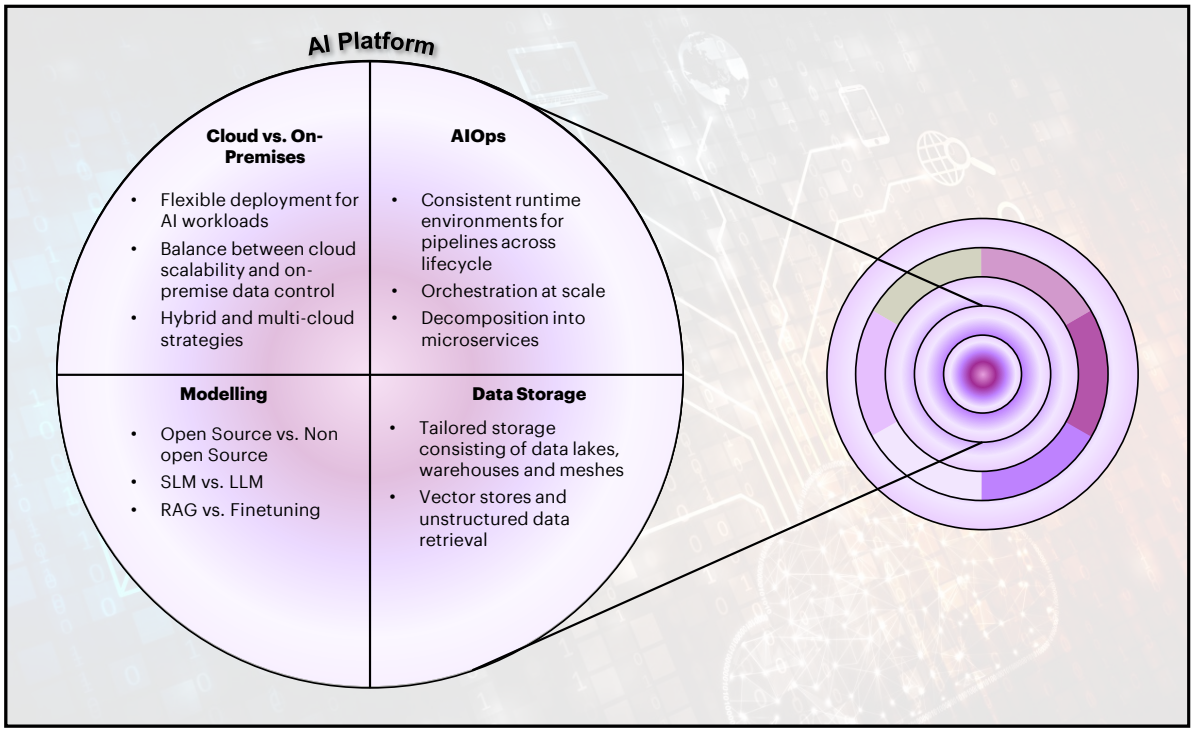


Figure 9: Pillars of AI Platforms in Engineering AI Landscapes

1. **Cloud vs. On-Premises:** For engineering enterprises aiming to leverage AI at scale, the choice between on-premises and cloud-based platforms is more than a technical decision. It is a strategic consideration with long-term implications. AI workloads, particularly in product development contexts, are characterized by high data volume, velocity, and heterogeneity. As Theodorakopoulos et al. (2024) highlight, many on-premises infrastructures struggle to scale effectively under these conditions. In contrast, cloud environments offer dynamic scalability, enabling enterprises to process and analyze large, diverse datasets without investing in hardware expansion. When

assessing the trade-offs between cloud and on-premises deployments, enterprises should consider the following dimensions:

- **Security & Compliance**: On-premises solutions offer more direct control, while cloud providers offer certified compliance solutions but require trust in external parties.
- **Cost**: On-premises solutions require significant upfront investment and maintenance. Cloud follows a pay-as-you-go model (IaaS, PaaS), which scales with usage but may become costly over time.
- **Performance**: On-premises solutions can reduce latency for real-time systems. Cloud offers powerful hardware on demand but may suffer from network-induced delays.
- **Scalability & Availability**: Cloud platforms provide elastic scalability and built-in redundancy. On-premises environments are slower to scale and require manual fault tolerance measures.

A common concern among enterprises is the risk of vendor lock-in when committing to a single cloud provider. To mitigate this, enterprises can follow a multi-cloud approach that distributes services across different platforms, increasing resilience and operational flexibility (Dai et al. 2025). Moreover, the emergence of hybrid architectures, combining on-premises servers, edge computing, and cloud platforms, provides engineering companies with new levels of freedom to align technical needs with strategic priorities.[2] One of the most promising paradigms in this context is Federated Learning (FL), which allows AI models to be trained across distributed data sources without centralizing the underlying data. Yao et al. (2022) and Zhan et al. (2025) describe FL frameworks in which sensitive data remains on-premises or at the edge, while only encrypted model parameters are exchanged with the cloud. This method respects data sovereignty, minimizes bandwidth usage, and enables the training of models across heterogeneous environments. It is particularly effective in addressing latency, computational constraints, and system reliability.

In practice, the decision between cloud, on-premises or hybrid infrastructures should reflect the nature of the AI workload and the strategic priorities of the organization. On-premises implementations are best suited for environments where AI is tightly coupled with proprietary hardware and low-latency, real-time inference is essential, or where data sensitivity (e.g. in highly regulated industries) prohibits external transmission. Conversely, cloud-based solutions are recommendable for enterprises that need to support variable workloads and rapidly train and deploy new models.

2. **AIOps:** To successfully adopt AI at scale, engineering companies must go beyond isolated AI use cases and build robust infrastructures and architectures (see Figure 8) for development, deployment, and monitoring. AIOps, which represents the fusion of AI and DevOps, provides exactly that foundation. AIOps enables scalable and secure ML workflows, offering automation, standardization, and traceability across the entire lifecycle. While not yet widely implemented in industry (Faubel & Schmid 2024), AIOps will be essential for engineering companies seeking to operationalize AI effectively. Rooted in CI/CD principles, AIOps emphasizes microservices, containerization, and orchestration to ensure modularity, scalability, and reliability (Kreuzberger et al. 2023). It also plays a crucial role in protecting against security risks such as data leakage,

[2] See Loconte et al. (2024) for a comprehensive framework on hybrid industrial AI architectures involving IoT, edge, and cloud layers.

poisoning attacks, and systemic vulnerabilities (Dai et al. 2025). In practice, however, AIOps adoption varies significantly. Different levels of technical expertise, even within the same organization, influence how workflows are designed and operated (Rzig et al. 2024). Success depends not just on tools, but also on organizational alignment, training, and culture (Mehmood et al. 2024; Faubel & Schmid 2024).
With the rise of GenAI and LLMs, LLMOps has emerged as a specialized extension. Compared to standard ML or DL deployment, LLMOps must address greater compute and storage requirements, while ensuring scalability and responsiveness (Pahune & Akhtar 2025). Although still at an early stage of maturity (Borovits et al. 2025), companies are advised to invest early in LLMOps expertise to stay ahead of the curve, since AIOps and increasingly LLMOps are becoming strategic capabilities for engineering organizations aiming to industrialize AI development and deployment.

3. **Model Selection:** The selection of suitable AI models for engineering applications is both a technical and strategic decision. Engineering tasks are inherently diverse, from text- or code-based artifact generation (requirements, test cases & scripts, release notes) to 3D data processing and generation for CAD or simulation models. An overview of (selected) commonly used AI models in engineering contexts is presented in Table 1. In the last years, transformer-based LLMs have emerged as powerful tools capable of interpreting, generating, and reasoning over multimodal engineering data, including text, code, images, and time series. These models increasingly act not as isolated systems, but as components within Agentic AI ecosystems that collaborate to solve complex engineering tasks.

Table 1: Selected AI Models Typically Applied in Engineering Use Cases

Model	Description	Engineering Use Cases
Large/Small Language Model (LLM/SLM)	Transformer-based models capable of understanding and generating natural language	Requirement generation Test case generation Release note generation Text-2-CAD
Convolutional Neural Network (CNN)	Deep learning model specialized in processing high-dimensional data such as images or time series	Defect detection in XiL testing CAD/simulation model classification
Graph Neural Network (GNN)	Deep learning algorithm designed to operate on graph-structured data	BOM analysis Dependency and traceability analysis of RFLPT[3] artifacts
Physics-Informed Neural Network (PINN)	Neural networks that incorporate physical laws (e.g., partial differential equations) as constraints	Surrogate modelling for FEM/CFD simulations
Conventional ML algorithms (classification)	Different conventional ML algorithms such as random forest or support vector machines for classification tasks	Artifact property classification Duplicate or inconsistency detection
Conventional ML algorithms (regression)	Different conventional ML algorithms such as multilayer perceptron, decision tree or support vector regression for regression tasks	Parameter prediction based on CAD/simulation data Effort and performance estimations

[3] Requirements, Functional, Logical, Physical, Test.

As the codification of engineering knowledge accelerates, the role of LLMs and multi-agent systems is expected to grow. The future lies in orchestrating collaborative reasoning chains between (cloud-based) LLMs and on-premises-deployed Small Language Models (SLMs). In this paradigm, LLMs handle abstract, high-level reasoning tasks while SLMs execute context-specific operations at the device level. This hybrid architecture not only reduces latency and cost but also aligns with the distributed nature of industrial systems (Li et al. 2025d). Model size plays a crucial role in this division of labor. LLMs, with billions of parameters, offer generality and adaptability across domains. SLMs, by contrast, typically range from a few million to several hundred million parameters and are optimized for edge deployment on on-premises hardware, mobile devices or microcontrollers. As shown by Subramanian et al. (2025), while SLMs may lack the broad generalization capabilities of LLMs, they often outperform them in narrow, domain-specific tasks due to their efficiency, lower computational demands, and reduced inference costs. This suggests that smaller models are not a compromise, but a strategic advantage when deployed appropriately.

Beyond technical trade-offs, organizational considerations are increasingly shaping model selection. Issues of compliance, control over proprietary data, and risk management are prompting many companies to turn to open-source models that can be deployed on-premises and fine-tuned to meet domain-specific needs. One prominent example is DeepSeek, an open-source LLM that has demonstrated competitive performance with leading proprietary models like OpenAI's GPT or Google's Gemini, particularly in specialized tasks. DeepSeek not only allows fine-grained customization but also supports efficient domain adaptation (Guo et al. 2024; Rahman et al. 2025). Injecting domain-specific knowledge into AI systems remains a top priority for industrial users (Lee & Hu 2023). This challenge raises a fundamental decision point:

> *Should companies rely on out-of-the-box general-purpose foundation models, apply transfer learning for task-specific fine-tuning, build Retrieval-Augmented Generation (RAG) pipelines to supply contextual data dynamically, or train models from scratch?*

In most industrial scenarios, end-to-end training of LLMs is not a viable option due to data scarcity and high resource demands. As such, hybrid strategies combining RAG, fine-tuning, and specialized SLMs deployed on-premises represent the most promising way forward. In principle, each use case should be analyzed in detail to determine which model is best suited to the specific problem and in many cases, the use of LLMs can be avoided. Some engineering applications can be effectively implemented using other ML or DL algorithms that require significantly less computing power and lower data volumes.

4. **Data Storage:** Modern engineering departments produce large amounts of heterogeneous data, encompassing structured formats such as simulation outputs and measurement data, as well as unstructured sources like design documentation and research notes. Efficient polyglot storage architecture is essential to unlock the potential of such data for analytics, monitoring, and AI-driven decision-making. Traditional architectures like data warehouses and data lakes have served distinct purposes. Data Warehouses are designed to integrate structured, cleaned, and pre-processed data using ETL pipelines, enabling consistent reporting and historical analysis. In contrast, Data Lakes ingest both structured and unstructured data in its raw form via ELT pipelines, deferring transformation to query time and thus offering more flexibility for diverse analytics tasks (Azzabi et al. 2024). However, enterprises are increasingly facing a trade-off between the structured reliability of warehouses and the

flexible access patterns of data lakes (Dai et al. 2025). This has led to the emergence of the lakehouse paradigm, which combines the advantages of both models. As described by Armbrust et al. (2021), lakehouses retain low-cost, open file formats and avoid data duplication and staleness, while supporting SQL-based analytics and AI workloads within a unified platform. Lakehouses aim to simplify complex architectures by consolidating batch and streaming data pipelines, minimizing technology heterogeneity, and eliminating redundant data movement between systems (Schneider et al. 2024). They must meet rigorous requirements: consistent storage formats, support for CRUD[4] operations across all data types, relational tabular structures, a declarative query language, consistency guarantees, and task-isolated processing. Moreover, direct data access and unified batch-stream processing capabilities are crucial for enabling advanced AI workflows.

In today's increasingly complex engineering tool landscapes and AI platforms, the concept of polyglot storage architecture is becoming a necessity rather than a choice. As outlined by Kasper et al. (2024), storing all product lifecycle data in a single, monolithic database is neither scalable nor efficient. Instead, polyglot persistence enables a multidimensional representation of product data across various views (RFLPT), while optimizing performance, scalability, and data accessibility. This is in line with the decentralized data architecture based on the data mesh concept (Goedegebuure et al. 2024). Corresponding databases at domain and tool level thus represent the authoritative single source of truth (Bone et al. 2018, Kwon et al. 2020) and allow the higher levels (AI platform, context modules) to access this data in real time.

In summary, a centralized AI platform is essential for engineering enterprises to efficiently develop, deploy, and monitor AI use cases by providing scalable computing, storage, and containerized environments integrated into the overall enterprise architecture. Its success depends on robust design decisions across four pillars ensuring scalability, security, and effective AI adoption in complex engineering landscapes.

Industry Insight into Building a Scalable Cloud Data Platform

The client, a large industrial manufacturer, had factory sensors generating valuable data that remained siloed in a fragmented on-premises setup, creating bottlenecks and slowing innovation. Growing data volumes overwhelmed legacy systems, making it difficult to scale analytics efficiently, prompting the adoption of a cloud-based, scalable architecture for real-time data processing and AI-driven use cases like predictive maintenance.

The new data platform, built on a modular data lakehouse design, integrated open-source technologies such as Apache NiFi, Kafka, Spark, Airflow, Trino, and S3 object storage, all orchestrated on Kubernetes. This architecture supported elastic scaling, reproducible deployments, and unified access to sensor and enterprise data, providing a strong foundation for operational analytics and AI workloads.

The implementation proved that a modular, cloud-native architecture can deliver agility and scalability for industrial AI. Key lessons included the need for strong data quality management, clear data contracts, and close collaboration between data and operations teams. Remaining challenges include optimizing real-time inference workloads and improving end-to-end observability across data pipelines.

Guido Schmutz
Senior Data Architect
Accenture

Dr. Martin Wunderli
Head of Modern Data Platforms
Accenture

[4] CRUD: Create, read, update and delete.

04 Context Management

To fully exploit the potential of AI in product development, it is necessary to provide models with holistic access to diverse product information, such as product structures, development artifacts, documentation, development processes, legal requirements and standards and system models. Providing context and causal cause-and-effect relationships between the engineering domains involved in the development process is a key challenge. This challenge can be solved by creating and continuously maintaining context modules, such as graph databases (Kwon et al. 2020, Liang et al. 2024b), vector databases or meta-models. Both in engineering and in subsequent phases of the product life cycle such as manufacturing (Zhou et al. 2024a, Yahya et al. 2024) and service/maintenance (Xie et al. 2024), considerable research efforts have been undertaken in recent years to demonstrate the suitability of knowledge graphs for knowledge linking at an elevated meta-level. Product-related information that was originally available in unstructured formats such as documentation, emails, or regulations can now be stored in vector databases and made accessible to LLMs for RAG applications (Xu et al. 2025c). For end-to-end application of AI in the product development process, it is important to provide system-wide context and formalize the product development process in such a way that cross-domain processes, dependencies, and interactions become machine-readable. The definition and stringent application of semantics and ontologies, as well as the transformation of a document-centered development process toward MBSE (Zhang et al. 2025e), formalize product development and are prerequisites for the scalable application of AI across domain boundaries.

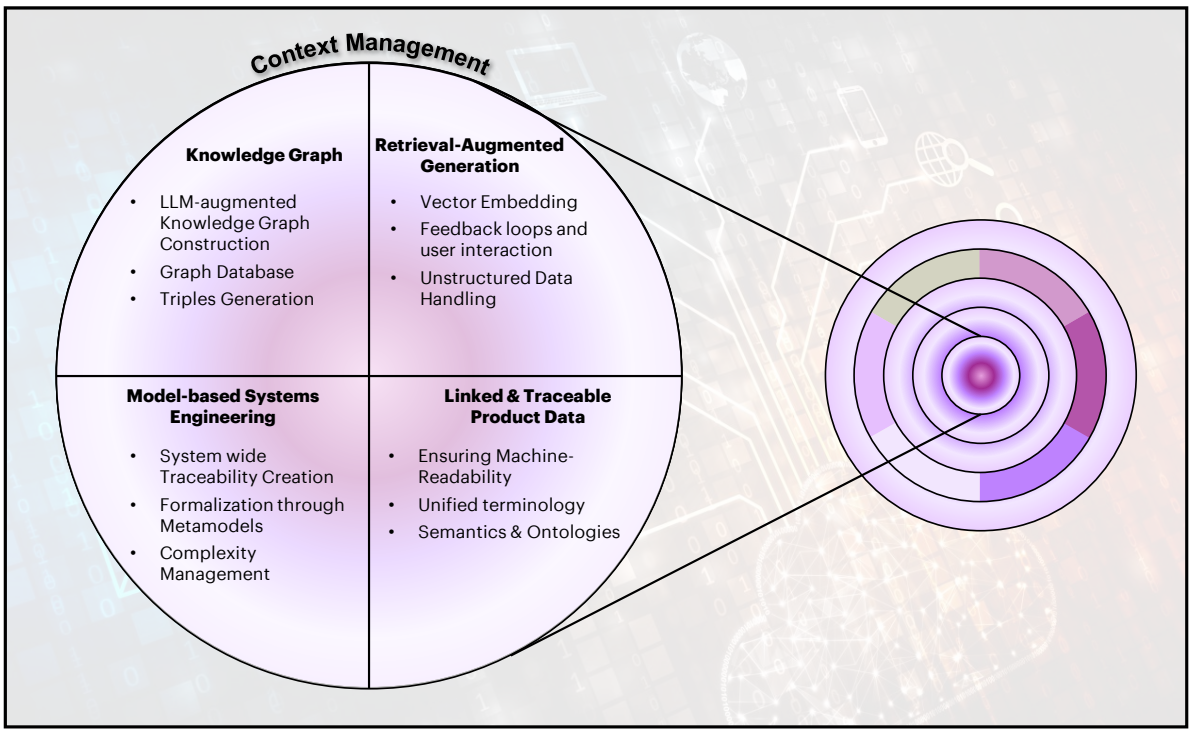


Figure 10: Pillars of Context Management in Engineering AI Landscapes

In the framework shown in Figure 3, context management serves as the brain of AI applications, as it links product knowledge and maps the interdependencies and impact chains of artifacts and other product-related information. Context management represents a control layer at metadata level and aims to provide context and cause-effect relationships between artifacts and product information, drastically improving the knowledge extraction, reasoning and orchestration capabilities of LLMs. It is therefore recommended that the support of vector and

graph databases is considered when selecting the AI platform and corresponding storage platforms (Harby & Zulkernine 2025). Figure 10 shows the pillars of context management in engineering AI landscapes, which are explained in more detail below.

1. **Knowledge Graphs:** Knowledge graphs consist of nodes (entities) and edges (relations) that structure knowledge in the form of subject-predicate-object triples (Liang et al. 2024a). They integrate information from various sources, semantically link related concepts, and enable context-aware queries and inferences. Ontologies are often used to formally define the meaning of nodes and relations, allowing machines to interpret the data (Zou 2020). While until a few years ago a lot of manual work was required by experts to create domain-specific KGs (Hur et al. 2021), generative approaches and especially LLMs, are now able to process big data and automatically create (multimodal) KGs based on heterogeneous data sources (Ibrahim et al. 2024). The construction of KGs in engineering is particularly challenging, as domain knowledge such as engineering principles or industry-specific best practices and regulations must be embedded in the KG and the underlying data is stored in a wide variety of source systems (Liang et al. 2024b, Liang et al. 2025). Furthermore, various measures must be considered in the KG construction phase, such as efficiency (computing time required), costs (number of tokens used) and KG quality (e.g. proportion of isolated entities) (Xiao et al. 2025).

   Figure 11 shows an exemplary LLM-augmented KG construction and retrieval pipeline as well as a simplified KG specialized for engineering. KG construction is based on heterogeneous data sources that draw on artifacts from the engineering toolchain, but also use additional information from documentation, regulations, internal wikis and many other data sources. Well-maintained metadata that documents the validity of the extracted information and artifacts for specific products, configurations and variants is fundamental to extract valid relations for the KG construction. In the first step of the KG construction pipeline, data is transformed, and relevant text sections are extracted. The extracted text sections are then further processed by an LLM in the LLM-augmented KG preparation step, which has the task of recognizing and extracting relevant entities and relations between entities and merging them into valid triples. This creates the simplified product KG shown in Figure 11, which depicts the relationships between engineering artifacts and further product information.

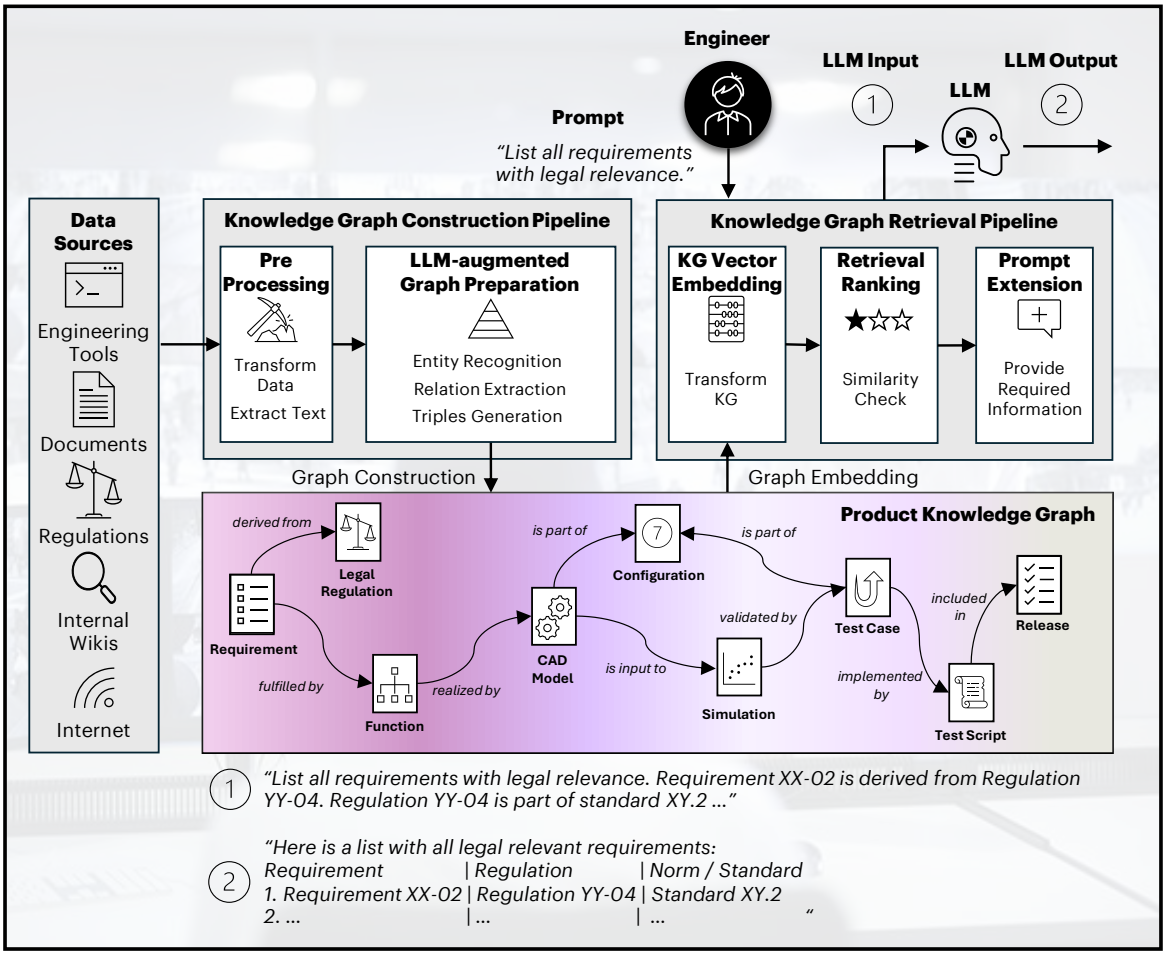


Figure 11: KG Construction and Retrieval Workflow based on Hoang et al. (2025) adapted to an Engineering Application

The constructed KG is then stored in a database. Relational databases are suitable for small graphs (e.g. Hoang et al. 2025), however when memory requirements increase, the switch to graph databases is unavoidable for performance reasons (Ibrahim et al. 2024). After the graph has been constructed, it is transformed into a vector embedding, making it accessible for LLMs and similarity checks with prompts. If an engineer sends a prompt to the LLM, the content of the KG is compared with the prompt, and the relevance of triplets is evaluated based on their similarity to the prompt. Triplets with high relevance are added to the prompt to provide the LLM with product and domain-specific context. Such an approach is called GraphRAG (Han et al. 2024b, Zhu et al. 2025, Peng et al. 2024) and is particularly effective for highly specific tasks where the LLMs need to draw on (domain-specific engineering) knowledge that was not provided to them during the training process (Zhang et al. 2025c). Different measures must also be considered for the retrieval pipeline. The most important measures are the time to create the vector embedding (indexing time) and the average retrieval time (Xiao et al. 2025).

Another promising approach is the use of DL algorithms to evaluate the relevance of triplets or combinations of triplets. GNN-RAG, i.e. the use of GNNs to identify highly relevant subgraphs and triplet combinations, enables LLMs to provide advanced reasoning capabilities (Mavromatis & Karypis 2024). The integration of agents that interact with KGs multiple times and optimize their actions based on reinforcement learning to retrieve the most suitable information from the KG (Luo et al. 2025) is a

further interesting research field. In the future, this can be a key for complex Product KGs that need to map different products, configurations, variants and the interaction of mechanics, hardware and software to identify complex relationships between artifacts and other product information, allowing conclusions to be drawn.

In addition to automated construction and retrieval, dynamic adaptations of KGs are an important feature to qualify for scalable use in engineering. This includes managing the KG and its vector embeddings in AIOps pipelines, which are dynamically adjusted as new engineering artifacts are created, and new product-related information is added (Liang et al. 2025). In particular, the introduction of Temporal KGs, which give information recorded in the KG a temporal reference and thus ensure the temporal validity of information (Choi & Jung 2025, Wang et al. 2024b), can be seen as a way of mapping change and configuration management processes in the future. The need for dynamic KG updates has already been recognized in the manufacturing sector (Wan et al. 2024) and prototypically implemented using the example of physically decoupled, collaborative robots (Bai et al. 2024). Another research focus is the continuous evaluation of the quality of KGs as well as the quantification of incompleteness and uncertainty. Existing (engineering) data is imprecise, incomplete and ambiguous, which is why these properties are also transferred to KGs. In addition, retrievals from KGs are also subject to uncertainty, which is why Mishra et al. (2024) call for the integration of uncertainty modules in KGs that highlight incomplete data areas, quantify knowledge gaps and then dynamically adapt the graph. The continuous assessment of uncertainties in retrievals from KG is also currently the subject of research (Ni et al. 2025a) and should be considered in future implementations in the monitoring layer of the AIOps pipeline.

Industry Insights into Fixed Entity Architecture for GraphRAG solutions

In developing performance optimized knowledge-based AI solutions, the initial approach utilized the LLMs based GraphRAG technique, specifically Microsoft GraphRAG. This approach demanded considerable effort to construct a graph, particularly with extensive data sets. It heavily relied on LLMs, lacked integration with domain ontology, and required substantial deduplication and post-processing. The main challenge was to devise a more cost-efficient, simpler, and production-friendly method that also improved domain comprehension.

The proposed method, named Fixed Entity Architecture (FEA), merges standard RAG with domain ontology. FEA employs a layered graph structure, where data layers are logically separated. While GraphRAG excels as a standard RAG approach by its nature, FEA simplifies the construction and utilization of graphs for specific GenAI applications. An extension of FEA, called NLP-driven GraphRAG, supports building layered graphs even in the absence of a predefined ontology. This involves text chunking similar to standard RAG, but with entities extracted and linked through triplets, thereby enhancing the traditional RAG model by incorporating entity relationships and logical connections.

The adoption of FEA resulted in a more robust and easier-to-develop graphs and query systems, improved domain understanding, and improved RAG performance through entity linkage and business domain logic. Feedback from implemented solutions showed notable gains in both efficiency and quality. Key takeaways highlighted the value of combining domain ontology with standard RAG methods and the effectiveness of NLP-driven GraphRAG in advancing the RAG framework.

Irina Adamchic
GenAI Expert & Graph Architect
Accenture

Bernhard Wieland
GenAI Expert
Accenture

2. **Retrieval-Augmented Generation**: To support RAG use cases, lakehouses are evolving to incorporate vector databases, which require high responsiveness and contextual

integrity. RAG's strength lies in making data accessible to AI systems that, just a few years ago, were largely not processable. Similar to the construction of KGs, unstructured data such as text is transformed into embeddings for RAG applications, which are then stored in vector databases. Using similarity searches between the prompt and the contents of the vector database, particular relevant information can then be extracted and incorporated into the prompt, providing LLMs and agents with context-rich information. This significantly increases the quality of LLM responses and prevents hallucinations (Zhao et al. 2024). Design decisions such as chunk size and prompt templating significantly affect retrieval quality (Li et al. 2025b), while system integration and failure point identification (Barnett et al. 2024) remain critical for operational reliability. Embedding RAG workflows in AIOps pipelines and connecting them to internal and external data sources ensures that lakehouses serve not only as repositories, but also as enablers of intelligent, context-aware applications.

RAG applications are now already common practice in the development of industrial AI use cases and scientific publications related to engineering. Numerous engineering AI publications have already demonstrated that providing domain-specific context via RAG and GraphRAG applications leads to more performant LLM applications. Examples span all engineering domains, such as requirements management (Masoudifard et al. 2024, Hey et al. 2025), architecture design (Hanke et al. 2025), CAD design (Xiong et al. 2025), software engineering (Strittmatter 2025), simulations (Pandey et al. 2025, Feng et al. 2025), test case generation (Wang et al. 2025a), product documentation (Tao et al. 2024, Pu et al. 2024), and compliance assurance (Sovrano et al. 2025).

3. **Model-based Systems Engineering**: MBSE represents the paradigm shift from document-centric engineering toward formalized, model-based approaches by introducing structured and standardized system models that enhance consistency, communication, and collaboration across domains and life cycle phases. Unlike traditional documentation, MBSE provides a unified modeling environment in which requirements, logical and physical architectures, simulation behaviors, and optimization objectives are coherently represented and continuously refined across domain boundaries (Zhang et al. 2025e). A major milestone in this evolution is the emergence of SysML v2, which introduces a new metamodel and textual notation designed for improved semantic expressiveness and interoperability across engineering tools (Vaicenavičius et al. 2025). SysML v2's standardized API enables seamless data exchange and interaction between domain-specific tools and the system model itself. This standardization allows external applications to query, update, or extend the SysML metamodel, creating the foundation for machine-readability and AI compatibility.

   Building upon this foundation, MBSE establishes a central, system-wide metamodel that links metadata to domain-specific specifications. This integration allows for continuous verification and validation (Cibrian et al. 2025), as well as automated compatibility checks throughout the engineering lifecycle. Such model-centric architecture paves the way for AI and agentic AI systems to gain a holistic understanding of the overall system. These AI agents can interpret system structures, processes, and boundary conditions, and autonomously delegate well-defined development tasks to subordinate agents operating within domain-specific tools or using domain-specific languages (see also Section *Outlook*). MBSE therefore provides a system-wide context that will enable AI applications to understand and explore the overall system in greater depth before development tasks can be orchestrated at subordinate system levels.

The integration of AI into MBSE has emerged as a research focus in recent years (Poulsen et al. 2025). Zhang et al. (2025f) outline a research roadmap that illustrates how GenAI can support model development, model management, and model comprehension. Early studies demonstrate that LLMs are already capable of generating SysML v2 models directly from textual descriptions, thereby automating parts of the system modeling process (Longshore et al. 2024; Johns et al. 2024). These developments highlight the potential of combining formalized system modeling with AI-driven reasoning to accelerate system design, ensure traceability, and establish the digital foundation for intelligent, adaptive engineering ecosystems.

4. **Linked and Traceable Product Data:** A significant portion of time is spent by engineers searching for product information relevant to their tasks (Chandrasegaran et al. 2013), which is why the demand for linked and traceable product data is higher than ever. Although modern PLM software offers sophisticated solutions for PDM (Eigner 2021), the solutions reach their limits as soon as the number of variants and configurations in the product portfolio becomes unmanageable (Failla et al. 2025) and information that is not managed in the PLM system needs to be embedded. For this reason, linking product data and creating traceability across the entire product development process or even product life cycle in the sense of a digital thread is of fundamental importance and offers immense efficiency gains. This requires ontologies and semantics that describe the interaction of product data in a standardized way (Failla et al. 2025) and form the basis for the construction of KGs (Ryś et al. 2024).

   KGs are already being used in engineering for this purpose, as evidenced by several publications. While some authors propose KGs for domain-specific linking and traceability tasks, e.g. the similarity assessment of CAD models within large CAD repositories (Bharadwaj & Starly 2022), functional classification of components (Ferrero et al. 2022), knowledge retrieval from existing design data for product ideation in early design phases (Cong et al. 2025), conversion of standards into machine-readable formats (Luttmer et al. 2021) or model management (Ryś et al. 2024) and model versioning (Wu et al. 2025) in systems engineering, other authors propose cross-domain applications. Hedberg et al. (2020) propose a lifecycle handler system that assigns an ID to each artifact across domains and links them together via a KG. The importance of informative metadata for each artifact is emphasized to describe the content of the artifacts in detail and identify links to other artifacts. Kwon et al. (2020) show how they link design (STEP format) and inspection data (QIF format) via a KG and thus establish traceability between product design and quality assurance. Kasper et al. (2024) propose a KG-supported concept for linking data from all phases of the product life cycle, which can accelerate cross-domain change and quality management processes in the future (Kommineni et al. 2024). We assume that scalable solutions will be developed in the coming years that address cross-domain data interconnection and will be based on the approaches described (KG, RAG, MBSE).

The need to link product data across the entire product life cycle has been known for years. Nevertheless, scalable solutions that link data across domains and tools have not been available. With the emergence of LLM-augmented KGs and associated retrieval and reasoning capabilities, this is likely to change in the coming years (Liang et al. 2025). The prerequisites for exploiting this potential are the creation of clear ontologies, (automated) creation and maintenance of metadata and the development of expertise in KG/RAG construction, retrieval, maintenance and uncertainty management. In this way, cross-domain development processes in particular, such as change, configuration and release management, can be massively accelerated and automated in the future. Emerging fields of research such as agentic context

engineering, i.e., the integration of agents to retrieve the optimal context from heterogeneous data sources (Zhang et al. 2025i), may further increase the performance of AI applications in the future and show great potential, especially for domain-specific tasks.

05 Federated Governance

AI is entering product development at speed, drafting requirements, generating design variants, accelerating simulation setups, and assisting integration and release (Paliwal et al. 2024). Yet organisations that scale beyond pilots share one trait: they treat governance as an engineering discipline, not just another review meeting. In fragmented tool landscapes and heterogeneous product domains, centralised control becomes a bottleneck, while laissez-faire creates risk, duplication, and drift. A federated governance model resolves this tension by combining autonomy for domain teams with shared, automated guardrails (Williams & Karahanna 2013).

Governance is the rule set of policies, processes, roles, and metrics that keeps data and AI assets aligned with business goals, regulation, and ethics (Otto 2011). In our framework, Data Sovereignty, control over residency, access, and usage rights, is not a parallel construct but a governance goal: a sovereign configuration of the overall system (von Scherenberg et al. 2024). Sovereignty ensures that high-value data continues to flow while remaining compliant with legal, ethical, and strategic constraints, binding datasets, models, and processes to provenance, entitlements, and declared purposes through enforceable, machine-readable policies. Figure 12 illustrates key elements of federated AI governance in product development and other AI systems.

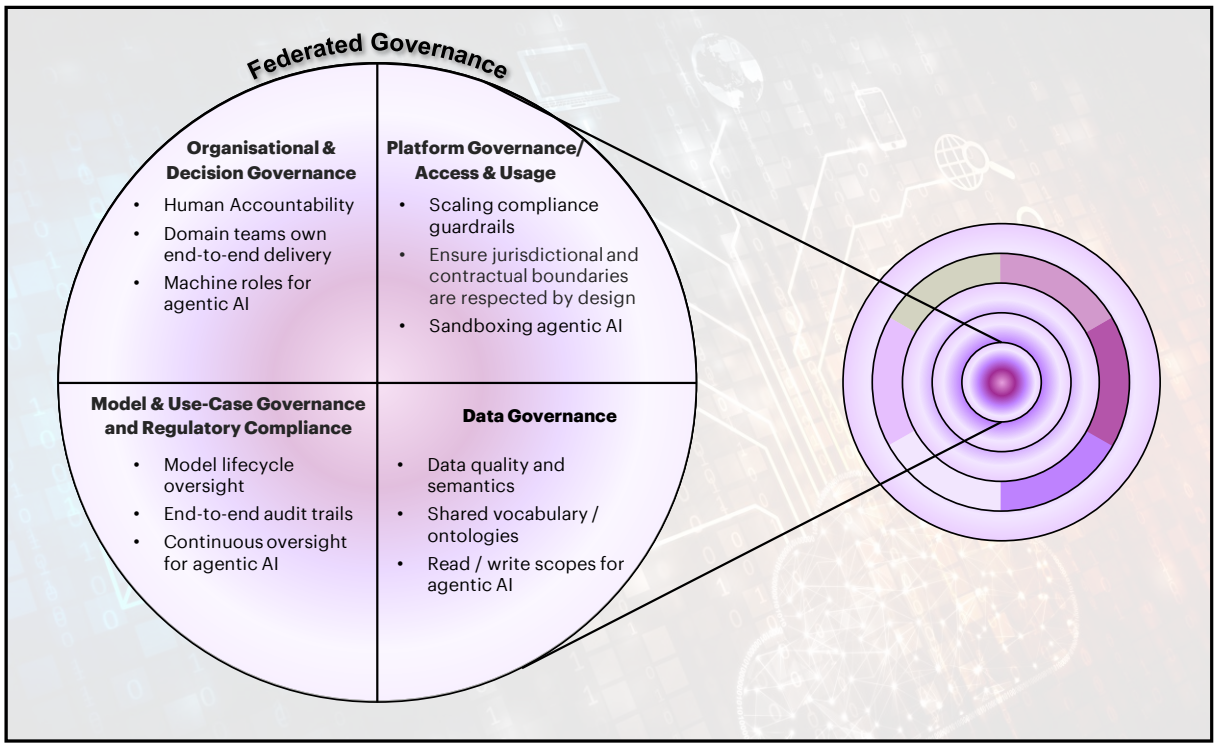


Figure 12: Pillars of Federated Governance in Engineering AI Landscapes

Federated governance spans the full product-development stack, as shown in Figure 3. At the core lies the context management, which captures lineage, usage, and audit evidence, serving as the metadata backbone for Data Governance. Here, sovereignty profiles and traceability ensure that every dataset, requirement, simulation, or release artefact is auditable and reusable.

Surrounding this core is the AI Platform, where Platform Governance enforces compliance by design through shared runtimes, orchestration, observability, and policy enforcement points. On top sits the Interoperability layer, providing the interfaces, APIs and communication protocols that link domains. This layer is shaped by Model & Use-Case Governance, which defines how prompts, models, and evaluations are integrated, monitored, and approved while remaining compliant with sovereignty constraints. The next layer *Data Quality* in the engineering toolchain is where domain teams create and curate data products, designs, and simulations, linking them back into the context modules under governance guardrails. Finally, the outermost layer of Federated Governance reflects Organisational & Decision Governance, the human system of roles, committees, and accountability that binds the stack together, ensuring that local autonomy across tools, data, and models remains aligned with enterprise-wide sovereignty, compliance, and business objectives. To operationalise this model, federated governance unfolds across four interdependent areas:

1. **Data Governance:** Data governance forms the foundation of any GenAI system by ensuring that information entering the AI lifecycle is reliable, traceable, and compliant with sovereignty rules. GenAI quality is bounded by data quality and semantics (Mohammed et al. 2025), thus, data governance encompasses the management of data quality, semantic consistency, classification, and retention, ensuring that only the necessary and permitted metadata are exposed to higher system layers. By embedding lineage and usage evidence in the knowledge graph, data governance establishes transparency and accountability at source. Sovereignty is realised through dataset profiles that encode access rights, residency, and sharing restrictions, automatically enforced through policy mechanisms. As GenAI becomes more agentic, data governance extends its scope to include autonomous data consumers and producers, defining their permissions and logging every access event as a traceable action. In doing so, it safeguards intellectual property, prevents data leakage, and enables responsible AI development across distributed domains.

2. **Model & Use-Case Governance and Regulatory Compliance:** Model and use-case governance ensures that GenAI models remain effective, compliant, and auditable throughout their lifecycle. It manages the intake and prioritisation of use cases, defines model cards and versioning schemes, and implements evaluation and monitoring for performance, bias, and drift. Risk tiering aligns with the EU AI Act, while Responsible AI principles (Accenture 2024)—human by design, fairness, transparency, explainability, safety, accountability, compliance, privacy, and sustainability—are operationalised through approval and monitoring processes. Sovereignty is maintained by checking prompts, fine-tuning datasets, and RAG pipelines against sovereignty profiles to ensure that every artefact used in model training or inference respects declared usage rights. End-to-end audit trails document who used what data for which model, when, and for what purpose. As AI systems evolve toward agentic autonomy, model governance extends to continuous supervision of agents' behaviours, defining approved roles, enforcing decision boundaries, and ensuring human-in-the-loop checkpoints with rapid rollback mechanisms (Kolt 2025). This way, it guarantees that innovation and autonomy coexist with accountability and compliance

3. **Platform Governance and Access & Usage Management:** Platform governance translates policy into infrastructure, embedding compliance and sovereignty enforcement directly into the technical backbone of AI operations. It defines and manages shared runtimes, container orchestration, vector databases, secrets management, observability, and policy-enforcement points at both build-time and run-

time. Through policy-as-code, it blocks non-compliant builds or deployments and enforces cost transparency, SLOs, and golden pipelines to ensure operational reliability. Sovereignty is preserved by enforcing data residency and segmentation through regional clusters and runtime isolation, ensuring that jurisdictional and contractual boundaries are respected by design. All access and usage are logged continuously, providing a verifiable record of platform activity. As AI becomes agentic, platform governance must include sandboxing, scoped authentication tokens, rate-limiting, and agent-aware observability—tracking decision traces and tool invocation logs to prevent privilege escalation and unauthorised lateral movement. By codifying compliance within the infrastructure itself, platform governance ensures that scalability and security advance hand in hand (e.g., Hurni et al., 2020).

4. **Organisational & Decision Governance:** Organisational and decision governance establishes the human and procedural scaffolding that ensures accountability, coherence, and ethical alignment across federated teams. It defines clear roles, responsibilities, and RACIs, creates lightweight but effective change-control processes, and introduces oversight bodies such as AI steering committees and ethics boards to manage high-risk approvals and cross-domain standards. Post-incident reviews, replay sessions, and continuous training in Responsible AI, privacy, and secure development maintain organisational readiness and trust. Sovereignty becomes a leadership KPI, with compliance rates, audit outcomes, and data-quality metrics linked to performance incentives. As AI agents increasingly collaborate with humans, organisational governance expands its scope to define machine roles and supervision rules—clarifying when must humans approve, override, or intervene and how escalations are handled when agents face uncertainty. Through this hybrid accountability model, organisational governance anchors the system in human responsibility while enabling autonomy at scale.

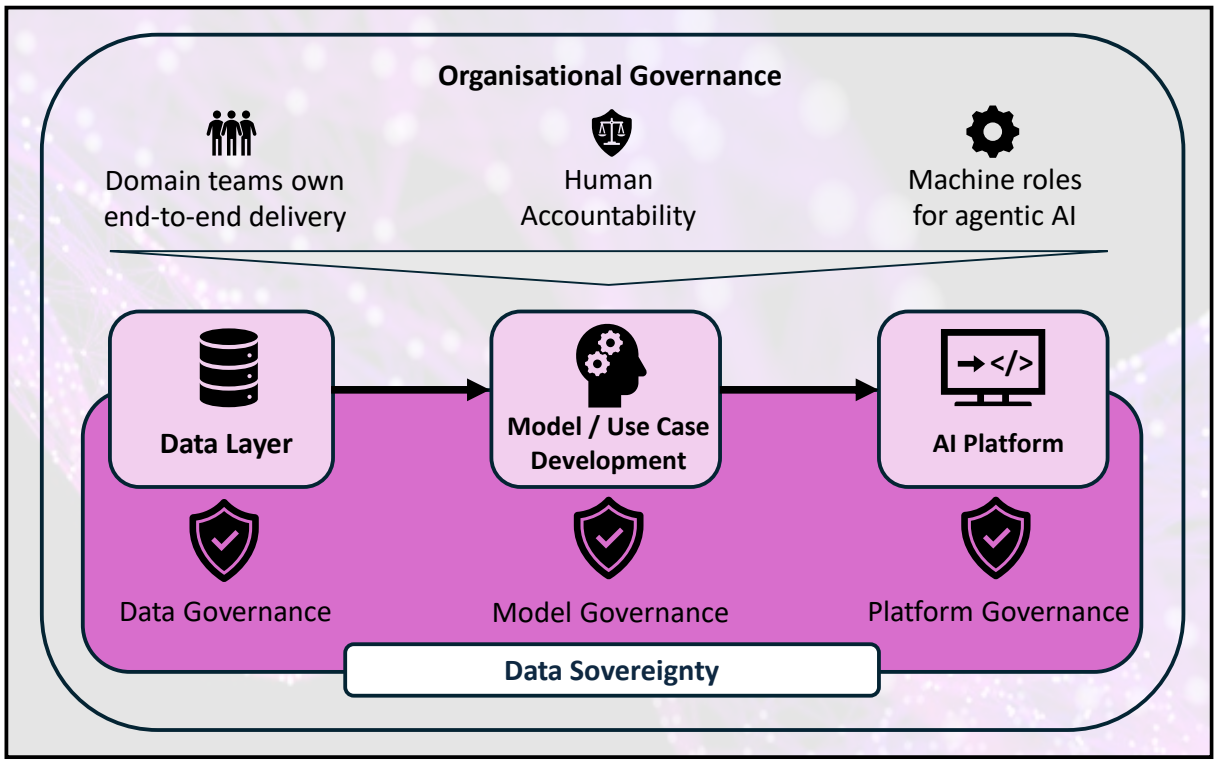


Figure 13: Governance Fabric for AI in engineering landscapes

The areas form a single governance fabric as depicted in Figure 13. Data governance supplies reliable, sovereignty-bound inputs to model governance, which sets policy requirements and risk posture that platform governance enforces automatically at build- and run-time. Organisational governance provides accountability and fast decisions when tensions arise (e.g., when a sovereignty profile restricts training, or a platform policy blocks deployment). The Product Knowledge Graph is the shared backbone, linking lineage, model usage, and platform logs including agent actions into one auditable view. These interconnections keep domain autonomy aligned with enterprise sovereignty and compliance objectives.

In industrial applications, where complexity, regulation, and global competition converge, governance across these four areas is infrastructure, not bureaucracy, the fabric that makes AI safe, scalable, and competitive. Without data governance, trust in inputs collapses; without model governance, bias and drift erode value; without platform governance, scaling breaks under complexity; without organisational governance, accountability diffuses, and sovereignty remains aspirational. Treating governance as an engineering discipline, which is automated, federated, sovereignty-driven, and agent-aware, is a competitive prerequisite for the GenAI economy.

Industry Insights into Model Governance

The EU AI Act distinguishes between different types of AI systems based on their risk level. High-risk AI systems (e.g., systems based on personal data or targeting critical infrastructure) are required to perform rigorous data governance activities to provide transparency and minimize risks. These requirements include ensuring that training, validation, and test data sets are relevant and accurate. For instance, they should avoid bias in the data set.

To simplify and support the data governance activities required by the EU AI Act, we utilized AI methods. Specifically, we implemented solutions for automatically generating and analyzing metadata to enhance data lineage and track the origin of data sets. Additionally, we implemented algorithms to detect potential biases (e.g., using techniques for identifying feature importance).

As part of our use case, we found that AI has great potential for addressing the data governance requirements posed by the EU AI Act. By combining multiple solutions for different governance aspects, the developers of high-risk AI systems can reduce the efforts for implementation.

Marcel Altendeitering
Head of Department
Fraunhofer ISST

Tobias Guggenberger
Group Lead
Fraunhofer ISST

Stages of AI Readiness in Engineering

The framework described in the previous Section (see Figure 3) represents the foundations and prerequisites for enabling scalable AI applications in engineering. To provide companies with guidelines for enabling AI in their product development processes, this chapter presents a maturity model that rates the maturity level of use cases with respect to the type of system integration.

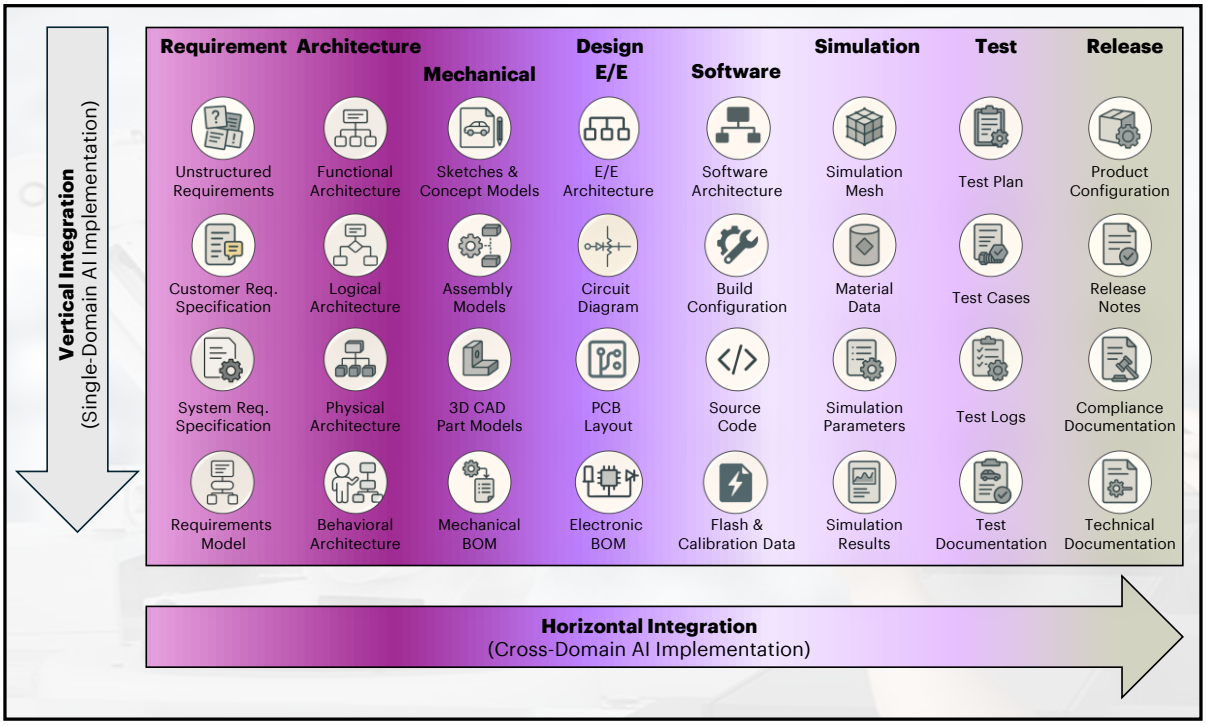


Figure 14: Horizontal and Vertical Integration in Engineering AI Use Case Implementations

Figure 14 provides an (illustrative, non-exhaustive) overview of the creation of engineering artifacts throughout the product development process. A distinction is made between two types of integration in systems engineering. Vertical integration describes the successive processing of artifacts within a domain (e.g., from unstructured requirements to system requirement specifications to detailed hierarchical requirement models), while horizontal integration describes the linking and traceability of artifacts across domain boundaries (Eigner 2021). These properties can also be transferred to AI use case implementations. Vertical use cases only access data from the domain that also uses the results of the use case. Horizontal applications access data from other domains, which requires traceability of artifacts throughout the product development process. To assess the maturity of enterprise AI applications, the automation levels of autonomous driving are proposed in the literature (SAE 2014). These divide the automation levels for AI applications in engineering into six discrete stages, ranging from fully manual engineering without AI use (level 0) to fully autonomous AI engineering (level 5) (Bernijazov et al. 2025).

Figure 15 shows the different levels for vertical (top) and horizontal (bottom) AI integrations and describes each level. The maturity levels of vertical integration refer to the degree of autonomy of AI applications regarding the performance of engineering tasks within a development domain. In contrast, the maturity levels of horizontal integrations refer to the degree of autonomy regarding the networking and traceability of artifacts from different domains. While

maturity level 0 does not involve any application of AI, the degree of autonomy increases successively with each maturity level until, at level 5, AI can perform domain-internal (vertical) and cross-domain (horizontal) tasks completely autonomously.

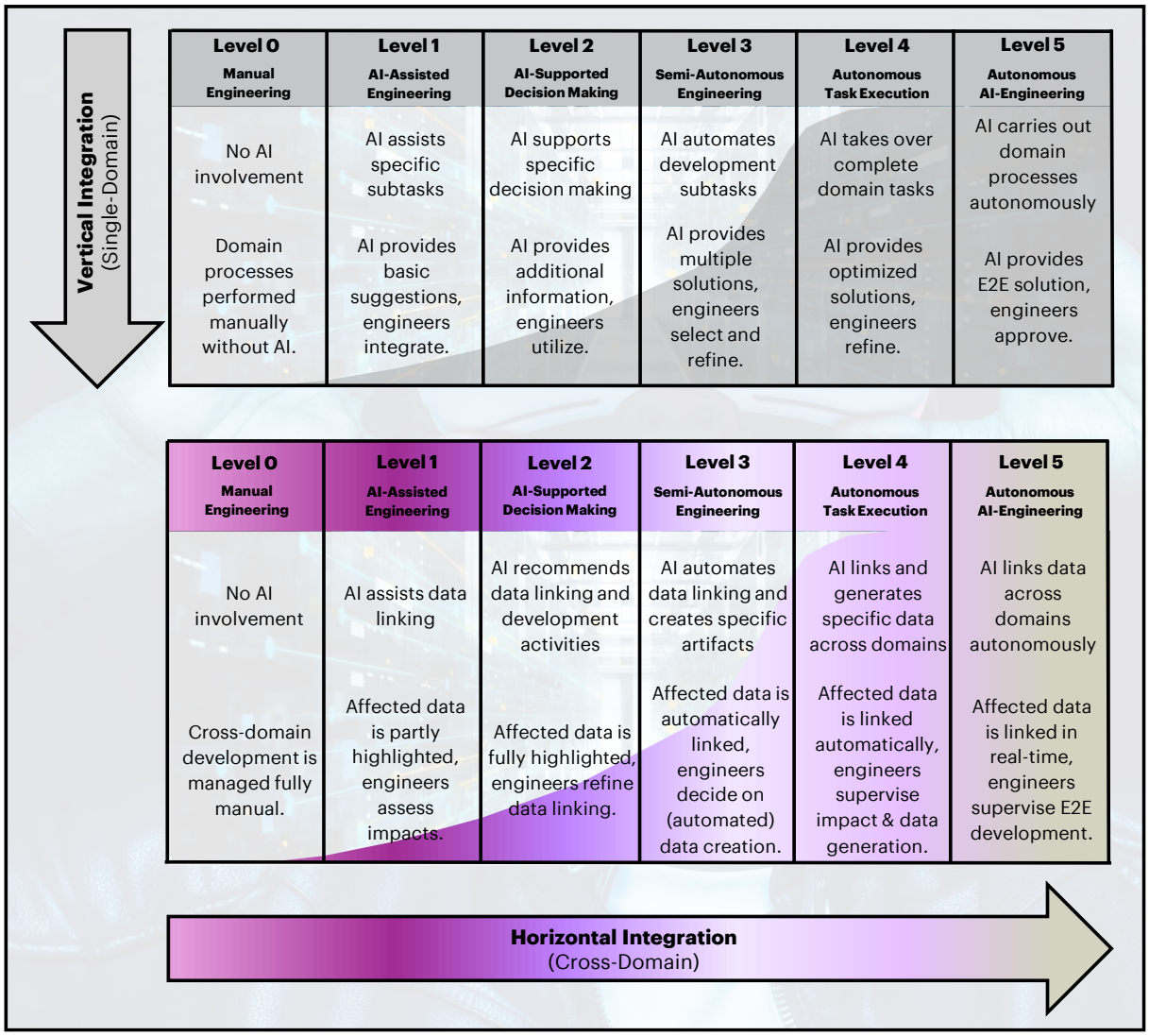


Figure 15: Stages of Vertical and Horizontal AI Readiness in Engineering

The automation levels serve as a basis for assessing the maturity of the use cases in the following Section. Various publications from the literature are analyzed and classified according to their vertical and horizontal maturity levels.

Use Cases

In this Section, we present several AI use cases across the six development domains in the V-model as well as cross-domain use cases. For each of the six development domains, three promising use case classes are presented that exemplify the current state of the art as reflected in the scientific literature. These examples provide a concise overview of the application of AI in the respective development domain and are intended to support C-level executives and engineers in identifying high-priority AI applications. In addition, key challenges are analyzed that currently hinder further increases in the automation levels, as defined by the maturity model shown in Figure 15. The selection of use cases does not claim to be exhaustive but instead deliberately focuses on high-value approaches that have already led to significant progress and innovation in literature.[5] The use cases are summarized at the end of this Section, cross-domain applications are presented, and they are evaluated according to their vertical and horizontal maturity levels in line with the previous sections (see Figure 15).

Figure 16 illustrates the V-model for the development of mechatronic and cyber-physical systems according to VDI (2021) and provides an overview of the engineering domains introduced in the following Subsections.

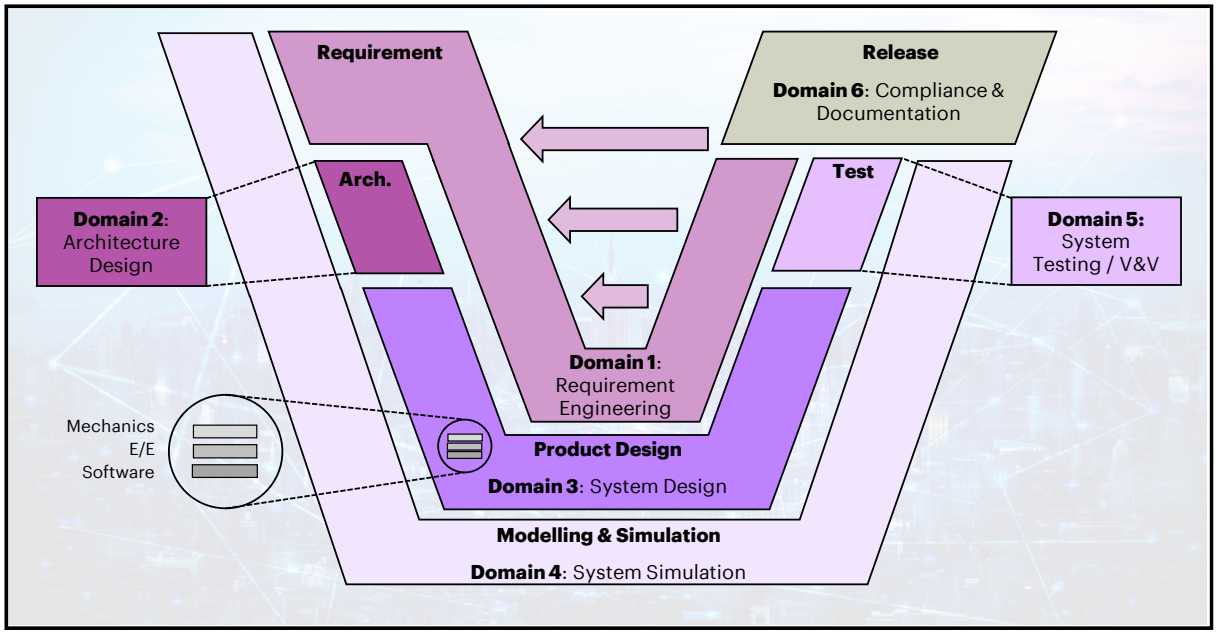


Figure 16: Overview of the six engineering development domains of the V-model

[5] Further engineering use case proposals and summarizations are presented by Bleisinger & Eigner (2025), Steffen et al. (2025) and Liang et al. (2025).

Requirement Engineering

The integration of AI, and specifically LLMs, into requirement engineering is transforming how engineering teams elicit, specify, and refine requirements throughout the product development lifecycle (Arora et al. 2024a). Traditionally a manual and therefore error-prone process, requirement engineering can benefit in the future from AI's capability to process vast volumes of unstructured and semi-structured information (Cheng et al. 2024).

AI models are capable of ingesting diverse inputs such as natural language requirements, technical documentation, programming code, examples, and sketches (Hemmat et al. 2025). From these, they can generate outputs that include (software) requirements and specifications, code and pseudocode, and models such as UML or SysML diagrams. The key benefits of applying AI in requirement engineering include enhanced quality and consistency of requirements, accelerated elicitation and management processes, improved traceability and connectivity between requirements, and the generation of artifacts that support downstream engineering tasks (Hemmat et al. 2025).

AI applications in requirements engineering can be grouped into three main categories of tasks as structured similarly by Hemmat et al. (2025) for hardware requirements and Norheim et al. (2024) for software requirements:

- **Requirement Generation**, where AI assists in drafting consistent and structured requirements and requirement models from unstructured inputs, such as stakeholder inputs or regulatory documents,
- **Requirement Optimization**, which focuses on evaluating and optimizing requirements for clarity, completeness, consistency, and compliance with formal language standards or domain-specific guidelines,
- **Requirement Analysis**, where AI is used to track dependencies, identify conflicts, and align requirements with each other and with downstream artifacts, such as MBSE models, functional and logical models or test cases.

Requirement Generation

One of the most immediate applications of GenAI is in the generation of requirements from unstructured stakeholder input, such as interviews, notes, or informal descriptions. By leveraging LLMs, this input can be transformed into well-structured, formalized requirement statements that align with engineering standards and stakeholder demands. This not only accelerates the elicitation process but also reduces the risk of overlooking critical stakeholder needs. Furthermore, GenAI can iteratively refine initial drafts through interactive dialogues, allowing stakeholders to clarify and validate requirements in real time. Ronanki et al. (2023) demonstrate that LLMs like OpenAI's ChatGPT are effective in eliciting functional and non-functional requirements through conversational prompts. Similarly, Nouri et al. (2024) show that safety requirement elicitation for autonomous driving systems can be significantly accelerated using LLMs, providing a structured and complete set of requirements faster than traditional manual methods. Voria et al. (2025) introduce RECOVER, a pipeline that structures stakeholder dialogue and drafts system requirements using Llama 2.

Requirement Optimization

AI supports optimization and quality assurance by detecting issues like ambiguity, redundancy, inconsistency, and syntactic errors. These models can evaluate requirements against predefined quality criteria and suggest rewordings that improve clarity, verifiability, and completeness. This leads to fewer misunderstandings and rework in later development phases. Additionally, AI can be used as a first-pass reviewer, enabling engineers to focus their manual reviews on higher-level content validation rather than basic linguistic or structural issues. Bashir et al. (2025) demonstrate how LLMs can be used to detect and explain ambiguities in requirements from the railway industry. Lubos et al. (2024) report that LLMs can reliably identify quality flaws in software requirements and suggest alternatives that are more precise, verifiable, and appropriate. Fantechi et al. (2023) further demonstrate that LLMs can detect internal inconsistencies across requirement sets, expediting the refinement process. Saleem et al. (2025) show that prompt-engineered LLMs can effectively classify requirements into functional and non-functional categories, improving consistency and traceability. Gärtner & Göhlich (2024) present an LLM-based approach to optimize automotive requirements regarding ambiguity, redundancy, consistency, clarity and compliance. However, these studies stress the necessity of expert validation and human-in-the-loop approaches to ensure reliability.

Requirement Analysis

Beyond textual interpretation, AI facilitates requirement traceability creation by extracting knowledge from requirements and making it usable for downstream engineering tasks. This includes identifying relationships between requirements, ensuring compliance with standards and regulations, categorizing them, and linking them to relevant models, design elements or test cases. As a result, complex requirement sets become more navigable across large-scale projects. Moreover, the generation of structured representations, such as knowledge graphs, supports traceability and consistency across different engineering domains and toolchains. Liu et al. (2025) introduce a method for building knowledge graphs from aerospace requirements using LLMs, which helps improve manageability and comprehension of complex systems. Similarly, Tikayat Ray et al. (2024) demonstrate how NLP algorithms can understand and map interdependencies between requirements in the aerospace domain. Hassine (2024) expands on this by showing how LLMs can be used to create traceability links between requirements and goal models. Using the example of software requirements, Masoudifard et al. (2024) show how specifications from regulations and standards can be considered to align compliance with corresponding software requirements.

Further publications, e.g. Fuchß et al. (2025a & 2025b), Niu et al. (2025), Hey et al. (2024 & 2025), examine the possibilities of LLMs for automated traceability creation and validation between requirements and other engineering artifacts. This includes

- the automated creation of system architectures in the context of MBSE (Akundi et al. 2024, Meng & Ban 2024, Bonner et al. 2024),
- the accelerated design of CAD models (Li et al. 2025a),
- the generation of simulation setups and parameters (Lebioda et al. 2025),
- the generation of software code (Han et al. 2024a)
- as well as the automated creation, execution and verification of test cases (Alagarsamy et al. 2024, Reinpold et al. 2024, Ferrari & Spoletini 2025).

These publications underline both the growing interest in research and the industrial relevance of requirement traceability along the product structure and along the product development process.

Despite these promising use cases, several challenges currently constrain the widespread AI adoption in requirements management.

- **Data-related challenges** include the limited availability of requirement-specific datasets, inconsistent annotation standards, and inadequately defined requirement engineering use cases (Norheim et al. 2024). These gaps hinder model training, benchmarking, and reliable evaluation across domains.
- **Methodological and organizational challenges** include the development of new requirement engineering practices to incorporate AI tools, explainability of AI-generated outputs, and the need for human-centric evaluation of requirements (Habiba et al. 2024). Additionally, the misalignment between AI developers and engineering end-users introduces further complexity in tool integration and practical application.
- **Technical challenges**, as noted by Hemmat et al. (2025), revolve around ensuring the completeness and quality of AI outputs, handling code and test generation effectively, managing input prompt design, and maintaining structured formatting. These issues directly impact the usability and trustworthiness of AI-generated requirements artifacts.

To address these challenges, dedicated research efforts and the development of industrial applications are needed. Methodological and organizational challenges require new approaches to effectively integrate GenAI into requirements engineering. Initial contributions in this area include the framework proposed by Ahmad et al. (2023) for human-centered AI-based requirements engineering, which emphasizes collaboration between engineers and AI systems. Complementing this, Vogelsang and Fischbach (2025) provide practical guidelines for applying AI to requirements engineering tasks, covering prompt design strategies, quality validation methods, and approaches for integration into existing development workflows.

If these challenges are overcome, GenAI can significantly increase the degree of automation in requirements engineering. Figure 17 shows the maturity levels of automated requirements management based on the automation levels.

Level 0 **Manual Req. Tracker**	**Level 1** **AI-Requirement Assistant**	**Level 2** **AI-Requirement Decision Supporter**	**Level 3** **AI Requirement Processor**	**Level 4** **AI Requirement Manager**	**Level 5** **Autonomous Req. Management**
No AI involvement	AI assists specific subtasks	AI supports specific decision making	AI automates development subtasks	AI takes over complete domain tasks	AI carries out domain processes autonomously
Requirement Mgmt. activities are performed fully manual.	AI supports engineers in formulating or rephrasing requirements.	AI provides insights in assessing completeness or identifying ambiguities.	AI classifies requirement clusters and provides templates for req. models.	AI generates compliant, consistent and traceable requirements.	AI generates and automatically adapts requirement models.

Figure 17: Vertical automation levels in AI-based requirements engineering applications

Industry Insights into Requirements Engineering

Automotive suppliers often receive many customer input documents per project with large volumes of text, ranging from 50 to 300+ pages of structured and unstructured text across multiple files that must be analyzed to extract and classify requirements. Normally, this responsibility is tasked to experienced systems engineers who manually read, identify, categorize, and map relevant requirements into the supplier's product hierarchy. Human engineers must read between the lines to identify requirements without keywords, or requirements irrelevant to the software domain such as those about paint composition. This process is not only time-consuming and mentally taxing but can also be prone to human error and inconsistencies if several engineers are working in silos. In one past client case, we estimated the effort required to be three engineers working full-time over three months to complete a single requirements input package.

Dr. Modar Horani
Managing Director
Accenture

To address these challenges, the Accenture team has developed an agentic AI approach leveraging Generative AI (GenAI) techniques built upon Natural Language Processing (NLP) and reasoning models to parse input documents, identify both new and duplicate requirements from an existing database, group related requirements, and flag requirements needing updates or clarification based on the newly extracted input requirements. We have prototyped this approach across several real-world datasets and consistently demonstrated drastic time savings, reducing a task that once took months to seconds. The results are also more consistent and less error-prone, providing a dependable baseline for engineering teams to refine further through their projects lifecycle.

Garrett Graham
Senior Principal
Accenture

A key insight from this use case was the implementation of the AI to reason beyond traditional keyword-based automation features of leading requirements management tools. The system now interprets project context against input document context, understands the semantic structure of requirements, and performs reasoning to assess relationships and redundancies which previously were only possible through human judgment. This represents a paradigm shift in how engineering organizations can scale quality and efficiency leading to decreased lead time to development with more time for innovation.

Architecture

The development of system and discipline-specific architectures is evolving with the integration of AI, which enables automation and augmentation of model creation, transformation, and analysis. One of the key benefits of AI in architecture development is the automated generation of system models from natural language requirements. AI and in particular GenAI can interpret textual inputs and propose initial SysML diagrams or block definitions, significantly accelerating the early stages of model creation and reducing manual effort (Bader et al. 2024). This capability not only speeds up the modeling process and ensures reusability of existing models but also helps ensure traceability from requirements to design artifacts and consistency in system development (Bernijazov et al. 2025).

AI applications in architecture development and MBSE typically process a variety of inputs, including requirements (Timperley et al. 2025), product design documentation (Zhang et al. 2025a), and detailed system specifications and architectures (Bernijazov et al. 2025). These inputs are translated into outputs such as SysML-based system architectures, functional models, and interconnected artifacts spanning across the engineering lifecycle. The advantages are multifold. Mottaghian et al. (2025) report significant efficiency gains and time savings, enhanced error reduction and quality assurance, the ability to establish and reuse engineering knowledge as a strategic resource, and improved human-centricity by allowing engineers to focus on higher-level, non-repetitive and creative tasks. According to Hovemann et al. (2025), AI use cases in architecture development and MBSE can be classified into three main categories:

- **Model Generation** focuses on the automated creation of system models from technical inputs, such as natural language requirements, interface descriptions, or technical documentations.
- **Model Optimization** focuses on AI evaluating and optimizing models for correctness, completeness, and compliance with modelling standards or engineering guidelines.
- **Model Traceability** involves using AI to link architecture artifacts with each other as well as with artifacts from other product development disciplines, and to retrieve information from them.

Model Generation

The generation of functional and architectural models is by far the most studied and practically implemented AI use case in the architecture domain. LLMs can automate the creation of SysML models and other formal representations based on unstructured or semi-structured inputs. For instance, Patel et al. (2024) and Timperley et al. (2025) both showcase how LLMs can derive SysML model entities from textual requirements. While Patel et al. (2024) focus on extracting model components from general unstructured requirement documents, Timperley et al. (2025) demonstrate a more domain-specific transformation of functional requirements for spacecraft systems into structured architectures consisting of functions, modes, and components.

Expanding into simulation and dynamic modeling, Zhang et al. (2025a) apply GenAI to generate executable models that represent the continuous dynamic behavior of aircraft electrical systems, starting from design documentation. A similar emphasis on structured model output is found in the work of Johns et al. (2024), who integrate a LLM into CATIA Magic to automatically generate conceptual SysML models for rocket systems within the design environment itself. In the work of Von Heissen et al. (2024), a plugin for Cameo Systems Modeler is developed that enables LLM-based generation of functional and logical architectures including SysML blocks,

interconnections, stereotypes, and diagrammatic representations for autonomous remote-controlled cars. Lameh et al. (2025) extend the application of LLMs beyond SysML by demonstrating the automated creation of feature models to support product line engineering, thus addressing variability management in system families. Bouamra et al. (2025) present a multi-agent system called SysTemp, that leverages LLMs to automatically generate SysML v2 models from natural language specifications, focusing on syntax correction and iterative refinement to address the lack of training data and improve model quality in systems engineering. Taken together, these studies illustrate the breadth of AI's potential in MBSE and discipline-specfic architecture development, from generating functional architecture and simulation models to create discipline-specific modeling artifacts, highlighting both the versatility and growing maturity of AI-assisted system modeling approaches.

Model Optimization

The assessment and optimization of existing models as well as the generation of alternative modeling variants is highly relevant in industry. AI holds significant promise in supporting engineers by identifying structural gaps, highlighting inconsistencies, and suggesting improvements based on learned modeling best practices. Moreover, it can assist in comparing and ranking alternative model structures or architectures with respect to predefined system goals, such as modularity, scalability, or fault tolerance. Such capabilities can help reduce modeling errors early in the development process and enhance model maintainability over time. In this context, Sultan & Apvrille (2024) present an AI-supported framework that leverages LLMs to detect inconsistencies in SysML models. Even though the number of publications on model optimization is not yet very high, we expect it to grow rapidly soon due to its high relevance to practice and the recent introduction of the textual SysML v2 notation.

Model Analysis

Another emerging application is the analysis of system architectures, particularly creating traceability across RFLPT artifacts and downstream design artifacts. As mentioned in the Subsection on requirement engineering, linkings between requirements, architecture objects, design drafts and test cases are crucial for end-to-end traceability along the product development process and can be seen as the basis for a cross-domain AI application. For example, Fuchß et al. (2025a) present an approach to link architecture documentation to architecture models using a RAG-based LLM application. Wawrzik et al. (2025) present a Knowledge Graph Generation Framework for Systems Engineering (KGG4SE), which automatically generates and quality-checks knowledge graphs from diverse sources, integrate them into MBSE tools, and thereby improve graph consistency, structure, and scalability. Karagoz et al. (2024) introduce a graph-based approach transforming SysML models into KGs. They apply a graph CNN to detect missing links, which addresses the problem of incomplete knowledge in MBSE system models and improves robustness, reliability and efficiency of complex system development. An AI-integrated framework for digital continuity and MBSE improvements is proposed by Xu et al. (2025a), focusing on enabling continuous feedback from early design and operational phases. Hanke et al. (2025) propose MBSE-Graph-RAG, a conceptual framework, which integrates knowledge graphs with RAG to enhance MBSE usability, accessibility, and automation by enabling natural language interaction, automated system architecture generation, and improved collaboration.

Further publications deal with discipline-specific automation solutions for electronic (Li et al. 2025f, Blocklove et al. 2023) and software architecture design (Esposito et al. 2025, Schmid et al. 2025).

Despite its promise, the application of AI in architecture development faces several challenges:

- **Data limitations:** There is a scarcity of structured, high-quality data specific to system engineering and discipline-specific architecture development, which hinders model training and generalization (Poulsen et al. 2025),
- **Adoption barriers:** Engineers require new skills to effectively use AI tools, including prompt engineering, model validation and evaluation. There is a steep learning curve and cultural resistance in some organizations.
- **Need for supervision:** While LLMs can generate comprehensive models, human supervision remains essential to ensure correctness, completeness, and adherence to engineering standards (Von Heissen et al. 2024; Timperley et al. 2025). Therefore, the integration of explainable and trustworthy AI is essential for architectural developments (Poulsen et al. 2025).

To address these challenges, Hovemann et al. (2025) recommend the development of optimized prompting techniques tailored for system engineering tasks. Additionally, Bernijazov et al. (2025) emphasize the importance of increasing the maturity of GenAI use cases step-by-step, starting with simpler tasks and gradually extending AI's role as confidence, trustworthiness and reliability improve. Figure 18 provides an overview of the maturity levels of AI-based architecture development applications and assigns core capabilities to the automation levels.

Level 0 **Manual Architecture Design**	**Level 1** **AI-Architecture Assistant**	**Level 2** **AI-Architecture Advisor**	**Level 3** **AI-Architecture Composer**	**Level 4** **AI-Architecture Generator**	**Level 5** **Autonomous Architecture Agent**
No AI involvement	AI assists specific subtasks	AI supports specific decision making	AI automates development subtasks	AI takes over complete domain tasks	AI carries out domain processes autonomously
Engineers define and maintain architecture manually.	AI suggests standard components, interfaces & templates.	AI proposes variants with (dis)advantages based on constraints.	AI creates high-level architecture drafts from requirements.	AI generates and validates architectural concepts for subsystems.	AI generates and adapts end-to-end architecture models.

Figure 18: Vertical automation levels in AI-based architecture development applications

Industry Insights into Agentic AI in MBSE

In complex, multi-disciplinary engineering environments, MBSE often plays a central role in defining product architectures, managing variability, and capturing cross-domain relationships. However, the growing scale and heterogeneity of digital engineering ecosystems expose a persistent limitation: insufficient integration between MBSE tools and domain-specific toolchains, such as requirements management, CAD/CAE, and simulation. Without strong bidirectional links, model updates and changes in one domain may not be reliably reflected in others, leading to data inconsistencies and reduced confidence in cross-discipline decisions.

To address these challenges, the Accenture team has integrated an Agentic AI system directly into the MBSE tool Catia Magic. This agentic AI approach combines AI expertise, in-depth understanding of MBSE methods and knowledge of the underlying engineering tools, a combination essential to meaningful context to AI-driven MBSE solutions. By understanding not only the model structures but also their engineering context (supplied by engineers and stored in RAGs), the AI can autonomously retrieve, interpret and relate data.

The agentic AI assists human engineers by automating repetitive modeling tasks, suggesting model updates, and highlighting inconsistencies, while human experts remain central in validation and decision-making. This has proven particularly effective in strengthening traceability across model elements and improving the consistency and completeness of variant configurations throughout the system lifecycle. A key insight from this initiative is that tight tool integration is essential for the successful deployment of Agentic AI in engineering environments. The ongoing introduction of SysML V2 will further accelerate this transformation, as its textual syntax and standardized APIs will enable much easier and more seamless integration of AI within MBSE ecosystems.

Dr. Christoph Schulze
Manager & MBSE Expert
Accenture

Martin Pauls
Systems Engineering Manager
Accenture

Product Design

AI has emerged as a transformative force in product design, particularly in CAD (Picard et al. 2025). It leverages a variety of inputs, including product specifications, part descriptions, sketches, and 3D CAD models, to produce optimized or entirely new designs in both two and three dimensions. Beyond generating geometry, GenAI systems are capable of learning structured design representations and similarity measures, enabling intelligent retrieval and modification of existing models. The benefits of applying AI in product design are multifaceted. According to the Accenture Research Report (2024), significant efficiency and time savings can be achieved. Gerhard et al. (2025) further emphasize that GenAI enables a human-centric design process by taking over repetitive tasks, thereby allowing engineers to focus on high-level conceptual and strategic design. GenAI also enhances creativity by supporting rapid idea generation and accelerates the evaluation of design alternatives through simulation or rule-based assessments.

However, the application of AI is not only desirable in M-CAD or E-CAD developments. Embedded software development in particular benefits from AI applications, as software code follows strict syntaxes of domain-specific languages and is therefore machine-readable. To address the complexity of modern development processes, we divide the Product Design Subsection into three parts, reflecting the three core disciplines of mechatronic product development. While mechanical design is carried out using M-CAD tools, the design of electrical and electronic components is performed with E-CAD tools. The rapidly growing importance of embedded software development, particularly in recent years, is also covered in this chapter and executed in various CASE[6] tools.

M-CAD Applications

The application of AI in mechanical product design can be broadly categorized into three major task domains, as suggested by Heidari & Iosifidis (2024):

- **Representation Learning**, which enables the extraction and structuring of design knowledge from existing models, including similarity measures and feature hierarchies
- **Model Optimization**, where GenAI assists in refining designs to meet specific engineering criteria
- **Model Generation**, where new design concepts or geometries are synthesized based on input constraints, specifications, or learned patterns from prior data.

Representation Learning

Representation Learning focuses on understanding and abstracting geometric and functional features from CAD models and retrieving existing, very similar designs as soon as possible. These learned representations serve as the foundation for a variety of downstream tasks. Jones et al. (2023) describe how models can learn meaningful features from CAD data to support advanced modeling and analysis. This includes classification of CAD parts based on geometric or functional criteria, segmentation of models into semantically meaningful components to support feature editing, and similarity analysis for retrieving comparable parts from large databases. According to Heidari & Iosifidis (2024), AI-based similarity retrieval can significantly support the creative process by surfacing existing designs that serve as alternatives or inspiration. Many publications show approaches on how the similarity to existing CAD designs

[6] CASE: Computer Aided Software Engineering.

can be evaluated automatically. These are based on GNNs (Quan et al. 2024), autoencoders (Jung et al. 2024) and unsupervised learning algorithms such as graph contrastive learning (Qin et al. 2025). Furthermore, Gao et al. (2024) present a weakly-supervised diffusion-based approach called DiffCAD, which retrieves and aligns CAD models from single RGB images. Representation learning and model retrieval have already been extensively researched, which is why the approaches presented here represent only a small selection of publications. For more in-depth review papers, please refer to Heidari & Iosifidis (2024) and Ning et al. (2025).

Model Optimization

In the domain of optimization, GenAI contributes to the refinement of design alternatives that are optimized for specific engineering criteria such as mass, stiffness, and stress distribution, while also incorporating manufacturing constraints. One prominent technique in this category is Generative Topology Optimization (GTO). Shin et al. (2023) provide an overview of how DL supports GTO using surrogate models that reduce computation time, handling high-dimensional inputs, learning of optimal parameters, and enabling exploration of broader design spaces. Qin et al. (2024) introduce an intelligent LLM-based system for shear wall structures that translates natural language into executable code, integrates generation with a two-stage optimization process, and accelerates design efficiency by up to 30 times while ensuring safety and cost-effectiveness. Major CAD software vendors like PTC (2024), Dassault Systèmes (2024), and Siemens (2024) have incorporated such optimization features into their platforms already, facilitating integrated simulation and validation workflows. Optimization GenAI applications are typically coupled with a simulation or a predictive AI model (see Surrogate Modeling use cases in Section Simulation) to evaluate the effect of the optimized modification on the engineering criteria (Kang 2025).

Industry Insights into Technical Drawing Assistant

The creation and validation of technical drawings and 3D models are essential for ensuring design integrity and compliance with engineering standards. However, inspection and review processes are often manual and time-consuming. Frequent revisions and the need for version control further increase the effort, while a lack of harmonization leads to inconsistencies and potential design errors. A major part of this complexity arises from collaboration between OEMs and suppliers, where communication, data handover, and alignment efforts are especially high.

To overcome these challenges, an automated inspection solution was developed that integrates 2D and 3D drawing checks directly into the engineering workflow. The system combines advanced image processing and rule-based validation techniques to automatically identify design errors and formal inconsistencies. By embedding the inspection process into the existing engineering environment, it enables automatic validation and minimizes design misinterpretations in cross-company interactions.

The result is a streamlined, standardized validation process that significantly reduces lead time and inspection effort for 3D checks. Beyond cost and time savings, the solution enhances data consistency across projects and supports better decision-making through automated reports and analytics. Ultimately, this approach transforms engineering validation into a continuous, data-driven process that ensures higher reliability and faster time to market. This leads to measurable quality improvements and stronger collaboration efficiency across the entire supply chain.

Christian Kohlschein
Associate Director
Accenture

Jann Pehle
Senior Manager
Accenture

Model Generation

Generation involves the creation of new designs based on minimal input, such as natural language, sketches, or point cloud data. The goal of generative models is to create 2D sketches or 3D CAD models based on the inputs.

In 2D sketch generation, Li et al. (2025c) demonstrate how stable diffusion models can generate car rim designs from basic prompts and subsequently transform them into 3D models. Liu et al. (2023b) explore how tools like DALL-E can generate product sketches from text, supporting early design ideation. Massoudi & Fuge (2025) compare the performance of a multi-agent and two-agent system for early-stage design of a solar-powered water filtration system. Both agentic approaches lead to valid JSON structures but only cover few requirements. In 3D model generation, several approaches exist. Badagabettu et al. (2024) show that simple text prompts can generate basic geometries, though increasing design complexity and sufficient quality requirements remain challenging. Xu et al. (2024) propose a multimodal model that integrates text, 2D images, and 3D point clouds to generate usable CAD geometries. Guan et al. (2025) present CAD-Coder, a system that incorporates reinforcement learning rewarding geometrical plausibility and syntactic correctness in the finetuning process. By leveraging a dataset of 110,000 triplets containing text prompts, CadQuery code (CADQuery 2024), and resulting 3D models, CAD-Coder achieves high-fidelity parametric model generation. Zhou et al. (2025) introduce CAD-Judge, which includes review modules to efficiently use LLMs for text-to-CAD generation, outperforming vision-language model-based methods in both accuracy and computational efficiency. Li et al. (2025e) develop LLM4CAD, an approach leveraging GPT-4 and GPT-4V for zero-shot 3D CAD generation from multimodal inputs, showing strong potential but revealing that text-only prompts often outperform multimodal ones except for complex geometries like gears and springs. A multi-agent framework is presented by Panta et al. (2025), who apply multi-modal LLMs to autonomously generate and iteratively refine parametric CAD models. Their framework consists of five agents (design expert, CAD script writer, executor, script execution reviewer, and CAD image reviewer) that work collaboratively together and generate CAD models by iteratively creating, executing, and refining scripts based on both textual prompts and visual feedback.

E-CAD Applications

The application of AI in electronic component design can be structured into three main E-CAD and Electronic Design Automation (EDA) application classes, as described by Pan et al. (2025):

- **RTL Design**, where AI supports creating Register-Transfer Level (RTL) descriptions in Hardware Description Language (HDL) such as Verilog or VHDL, defining dataflow, logical operations, and the circuit's functional behavior for synthesis.
- **Logic Synthesis and Physical Design**, where AI assists in transforming RTL descriptions into gate-level representations, optimizing placement, routing, clock trees, and power/ground networks to produce the final geometric layout.
- **Analog Circuit Applications**, where AI aids in selecting circuit topologies, sizing devices, and optimizing gain, bandwidth, and noise, including precise layout design for mixed-signal environments.

RTL Design

Ma et al. (2024a) introduce VerilogReader, a framework that integrates LLMs into the coverage directed test generation process to understand Verilog code and generate test inputs for uncovered lines or branches, significantly outperforming random testing for simple and

medium-level designs. Thakur et al. (2023) present AutoChip, a fully automated, feedback-driven approach that uses LLMs to iteratively generate and refine HDL code by leveraging feedback from Verilog compilers and simulations to identify and rectify errors. Tsai et al. (2024) propose RTLFixer, a novel framework designed to automatically fix syntax errors in Verilog code using LLMs, employing RAG and ReAct prompting to enable autonomous debugging with compiler feedback and human expert guidance. Chang et al. (2023) develop ChipGPT, a four-stage zero-code logic design framework that utilizes LLMs to automatically generate hardware logic designs from natural language specifications, demonstrating improved programmability and broader design optimization space. Collectively, these works demonstrate a strong trend towards automating critical and labor-intensive stages of RTL design, from code generation and testbench creation to error correction, by harnessing the advanced comprehension and generative capabilities of LLMs, often through iterative processes and structured feedback mechanisms.

Logic Synthesis and Physical Design

LLMs are increasingly being applied to enhance logic synthesis and physical design, streamlining various complex and time-consuming EDA workflows. Wu et al. (2024a) introduce ChatEDA, an autonomous agent designed to optimize the entire RTL to Graphic Data System Version II (GDSII) design flow through task decomposition, script generation, and task execution. To address challenges in EDA tool documentation Question-and-Answer, Pu et al. (2024) propose RAG-EDA, a customized RAG framework that leverages domain-specific techniques for better semantic understanding, reranking, and accurate answer generation. Similarly, Liu et al. (2023a) explore domain-adapted LLMs for industrial chip design with ChipNeMo, focusing on applications like an engineering assistant chatbot, EDA script generation, and bug summarization as well as analysis through specialized domain adaptation techniques. Chen et al. (2023b) present TRouter, a machine learning model-based framework for thermal-driven Printed Circuit Board (PCB) routing, which predicts thermal distribution to guide wire and via placement for lower-temperature designs. These studies highlight a focused effort to embed advanced AI capabilities into the later stages of chip design and physical implementation, with the goal of automating complex tasks, enhancing decision-making, and minimizing manual intervention. As with mechanical design, experience and data from previous product developments can be extremely important for such applications, as a great deal of implicit knowledge is contained in existing electronic designs.

Analog Circuit Applications

Analog circuit applications involve designing circuits that process continuous signals, such as amplifiers, filters, and converters. AI, particularly LLMs and GNNs, can significantly aid in automating and optimizing their complex design processes. For example, Chang et al. (2024) introduce LaMAGIC, a pioneering LLM-based topology generation model for automated analog circuit design, especially for power converter applications, which can efficiently generate an optimized circuit design from custom specifications in a single pass. Nau et al. (2025) propose SPICEAssistant, an LLM-based agent equipped with various tools to interpret feedback from the LTSpice circuit simulator and retrieve information from datasheets using RAG, demonstrating a significant improvement in the ability of LLMs to understand, adapt, and dimension electronic circuits. Plettenberg et al. (2025) present a GNN-based approach for automating the addition of optimizing components like pull-up/pull-down resistors, Resistor-Capacitor (RC) filters, and decoupling capacitors in PCB schematics to improve robustness and reliability by representing schematics as bipartite graphs and predicting component positions. Said et al. (2023) investigate the use of GNNs for circuit design completion in partially designed analog circuits,

where they identify missing components and predict their placement and connectivity within the circuit through a link prediction problem.

CASE Applications

The application of AI in embedded software development is poised to revolutionize the field, particularly with the advent of LLMs which demonstrate strong capabilities in understanding and generating code. As highlighted by Petrovic et al. (2025), industries with complex products such as the automotive industry can benefit heavily from AI adoption in the embedded software development, since stringent standards, hundreds of thousands of requirements and the trend toward software-defined vehicles lead to a huge amount of source code to be developed. Automating aspects of this development can significantly reduce human intervention and accelerate complex activities. In embedded software development, the application of AI, specifically LLMs, can be structured into three main classes:

- **Code Optimization**, where AI algorithms are capable of iteratively improving code for performance, fix bugs, and generate interpretable policies.
- **Code Generation,** where AI enables the automatic generation of executable code from diverse inputs like natural language requirements, formal specifications, or architectures, significantly boosting efficiency but demanding utmost precision and adherence to coding standards.
- **Code Analysis**, where AI analyses code to ensure compliance with safety standards and automatically creates traceability links to other development artifacts.

Code Optimization

Code optimization focuses on enhancing software quality, performance, and correctness. Ishida et al. (2024) develop LangProp, an iterative framework that optimizes LLM-generated code performance through data-driven feedback, particularly for autonomous driving policies. Sevenhuijsen et al. (2025) introduce VECOGEN, which refines LLM-generated C code using iterative formal verification feedback to ensure correctness for safety-critical systems. Kirchner & Knoll (2025) present a framework that optimizes AI-generated C++ code for automotive safety-critical systems via static verification and test-driven iterative refinement. These studies commonly emphasize iterative code refinement, guided by diverse feedback loops, to achieve high quality, correctness, and reliability, especially crucial for safety-critical applications.

Code Generation

Code generation in embedded software development harnesses LLMs to automate the creation of executable code from diverse specifications, significantly reducing development time and effort. Patil et al. (2024) propose the spec2code framework, which combines LLM-based code generation with formal verification to produce functionally correct, industrial-quality C code for critical embedded automotive software from diverse specifications, including formal ACSL. Liu et al. (2024a) empirically demonstrate the capability of GPT-4 to generate safety-critical C code for industrial domains, proposing a Prompt-FDC method that integrates functional, generalized domain, and constraint requirements to achieve high quality, completeness, and compliance. Nouri et al. (2025) developed a simulation-guided pipeline for LLM-based code generation, enabling iterative refinement and bug fixing of Python code for safety-critical automotive functions like Adaptive Cruise Control and Collision Avoidance by Evasive Manoeuvre based on feedback from virtual testing. Abdalla et al. (2024) explored automating the generation of MATLAB Simulink functions from software requirements for the automotive industry, leveraging fine-tuned open-source LLMs to create graphical programming code and documentation.

These works collectively highlight the potential of LLMs in automating code creation, particularly for safety-critical automotive applications, by emphasizing prompt engineering, iterative refinement, and integration with verification or simulation tools to ensure correctness and adherence to complex standards. Further publications extend AI's code generation capabilities to other specialized embedded software domains, including robotic controls (Luo et al. 2024), drones (Chen et al. 2023a), and microcontrollers (Haug et al. 2025).

Code Analysis

Code analysis focuses on understanding, verifying, and validating software code, ensuring its quality, correctness, and adherence to standards. Alturayeif et al. (2025) provide a comprehensive systematic literature review on machine learning approaches for automated software traceability, which is a crucial aspect of code analysis. Their work highlights how ML, DL, and LLMs are increasingly utilized to track and manage relationships between various software artifacts throughout the software development lifecycle particularly for safety-critical systems. This automated traceability, often formulated as a classification or ranking problem, links diverse artifacts like requirements, source code, and test cases, supporting essential processes such as change management, impact analysis, and quality assurance. The study emphasizes the growing adoption and superior performance of LLMs in this domain, while also addressing challenges like data scarcity and the need for standardized datasets.

Summary

Despite rapid advances, several challenges must be addressed for AI to become a robust part of product design workflows across the presented engineering disciplines.

- **Synchronization and parallel development across mechanics, E/E, and embedded software**: Modern systems engineering requires tightly coordinated development of mechanical, electrical/electronic, and embedded software components. However, these domains often follow different development cycles, tool chains, and maturity levels, making it challenging to maintain consistent design baselines and ensure traceability across disciplines (Berriche et al. 2020). AI-based design assistants must handle asynchronous updates, conflicting requirements, and cross-domain dependencies, while supporting continuous integration of design changes. Without robust synchronization mechanisms, the risk of design inconsistencies, late-stage integration issues, and costly rework remains high.
- **Insufficient data availability and model robustness**: The field continues to suffer from a lack of annotated and structured datasets, which limits the effectiveness of supervised learning approaches and necessitates reliance on unsupervised or self-supervised techniques. This data scarcity, combined with high variability in model performance across different design contexts, makes it difficult to ensure reproducibility and generalization of results (Heidari & Iosifidis 2024).
- **Challenges in human-AI collaboration**: The integration of AI tools into collaborative design workflows raises important questions about how designers and engineers should interact with AI systems. As Bordas et al. (2024) highlight, deeper research is needed to define effective roles, responsibilities, and interaction patterns between humans and AI in the creative process, particularly when dealing with complex design requirements and interdisciplinary teams.
- **Technical limitations in generating and structuring 3D content**: The automated creation of physically plausible and functionally valid 3D CAD models remains a significant technical hurdle. Guan et al. (2025) argue that overcoming this challenge

requires the development of unified datasets that link natural language prompts, generative code (e.g., CadQuery), and the resulting CAD models. Furthermore, to fully harness the capabilities of LLMs, it is necessary to make 3D data compilable in a software-like fashion, enabling the generation of interpretable and traceable model code that directly leads to valid 3D geometries.

Overcoming these challenges leads to higher levels of automation in the application of AI in product design, as illustrated in Figure 19, but still requires further research efforts.

Level 0 **Manual Product Designer**	**Level 1** **AI-Design Assistant**	**Level 2** **AI-Design Supporter**	**Level 3** **AI-Design Optimizer**	**Level 4** **Task-Autonomous AI-Designer**	**Level 5** **Autonomous Design Agent**
No AI involvement	AI assists specific subtasks	AI supports specific decision making	AI automates development subtasks	AI takes over complete domain tasks	AI carries out domain processes autonomously
Engineers rely on own expertise to generate product designs.	AI suggests standard design elements.	AI proposes multiple design options listing (dis-)advantages.	AI creates parametric models and runs optimization loops.	AI generates production-ready designs for specific parts.	AI manages end-to-end design across M-CAD, E-CAD and embedded SW.

Figure 19: Vertical automation levels in AI-based product design applications

Simulation

GenAI and related AI technologies are increasingly transforming simulation processes in product development, enabling faster, more efficient, and more adaptive virtual testing. The benefits of AI-accelerated simulation are substantial. According to the review by Herrmann & Kollmannsberger (2024), AI can be used not only to substitute traditional simulation with surrogate models but also to enhance simulations by replacing specific components within the simulation chain. These improvements lead to reduced computation times and lower resource requirements while maintaining acceptable levels of accuracy. Moreover, neural networks can be used to construct new discretization schemes, leveraging building blocks such as automatic differentiation, gradient-based optimization, and GPU-based parallelization. Generative approaches further expand the scope of simulation by synthesizing entirely new simulation scenarios or datasets based on learned patterns.

The application of GenAI in simulation can be broadly categorized into three major task domains:

- **Surrogate Modelling**, which focuses on replicating simulation outputs with lower computational effort applying AI.
- **Simulation Optimization**, where AI guides the tuning of design parameters within simulation loops.
- **Simulation Generation**, where new simulation scenarios, models or inputs are generated or multi-agent systems are applied to automate end-to-end simulation processes.

Surrogate Modelling

Surrogate modelling is an emerging application area of AI in engineering simulation, where machine learning models, particularly neural networks, are trained to approximate the behaviour of complex physical systems with significantly lower computational cost. These models act as stand-ins for traditional numerical simulations, enabling rapid prediction and design iteration. A fundamental distinction in this field lies between data-driven neural networks and physics-informed neural networks (PINNs). While data-driven models learn input-output mappings purely from simulation or experimental data, PINNs incorporate physical laws, typically in the form of partial differential equations, directly into the loss function by penalizing deviations from known physical behaviour. This hybrid approach improves generalization, particularly in data-scarce areas, and enhances the physical plausibility of predictions (Herrmann & Kollmannsberger 2024).

Recent research has demonstrated the versatility of surrogate modeling across a range of engineering domains. Hajisharifi et al. (2024) developed a reduced-order model that estimates critical simulation coefficients, drastically accelerating CFD simulation runtimes while preserving accuracy. Similarly, Jnini et al. (2025) presented a neural network-based, physics-constrained mapping from geometric configurations to flow field variables such as velocity and pressure, specifically applied to CFD simulations of curved backward-facing steps. Their work highlights how surrogate models can be tailored for complex fluid flow problems with geometrically sensitive dynamics.

In the structural mechanics domain, Sunil & Sills (2024) successfully predicted displacement fields in 2D FEM simulations using a PINN architecture, demonstrating that incorporating

physical constraints into learning enables accurate and generalizable surrogates for stress-strain analysis.

Simulation Optimization

Simulation optimization leverages the power of AI to automate and accelerate the tuning of input parameters in complex engineering simulations. By integrating AI with conventional simulation tools, this approach enables engineers to explore design spaces more efficiently, identify performance bottlenecks, and optimize system behavior with minimal manual intervention. A key component of simulation optimization is automated sensitivity analysis, which assesses how changes in input parameters affect simulation outputs. Traditionally a time-consuming task involving numerous simulation runs, this process can now be streamlined with AI models that learn the relationships between parameters and performance metrics. These models not only reduce computational overhead but also uncover non-obvious dependencies and interactions between parameters, enabling more targeted optimization strategies. In addition, parameter optimization is enhanced using generative models and intelligent search techniques. AI systems can propose candidate parameter sets based on learned patterns from historical simulations or desired output targets. Through iterative refinement, often guided by reinforcement learning or Bayesian optimization, these systems converge on optimal configurations that meet predefined objectives such as minimal energy consumption, structural integrity, or flow efficiency. Zhang (2025h) applies deep reinforcement learning algorithms to optimize turbulence model parameters, improving the accuracy and efficiency of CFD simulations. Zhang et al. (2025g) demonstrate how LLMs can act as decision-makers in parametric shape optimization for CFD simulations, efficiently guiding the search for optimal designs and outperforming classical optimization methods in convergence speed.

Further Publications that perform optimizations based on generative algorithms are already widespread in the literature, particularly for CFD simulations (Chen et al. 2024, Chen et al. 2025a, Pandey et al. 2025, Dong et al. 2025, Yue et al. 2025a, Yue et al. 2025b). The content of these papers is presented in the subsection on simulation generation, as they also involve the generation of the simulation setup.

Industry Insights into LLM-supported FEM Simulations

In the context of product development, Digital Twin models have become essential to meeting performance targets within realistic time and cost. However, their effective use often depends on specialized expertise concentrated in a few experts, creating a barrier to scaling the full potential of Virtual Product Development.

To address this, an Agentic AI approach was introduced, leveraging state-of-the-art LLMs integrated with a commercial Finite Element solver. The agents manage pre- and post-processing through Python APIs and coordinate High Performance Computing resources to execute simulations more efficiently. By automating these complex tasks, the system lowers the expertise required to interact with Digital Twin software, allowing engineers to dedicate more effort to product value creation instead of tool-specific activities.

The outcome was a significant improvement in efficiency, speed, and accessibility, enabling engineers with limited simulation experience to contribute effectively. A key insight from this initiative was the importance of coupling Agentic AI with robust simulation infrastructures and scalable data pipelines. Only within such an ecosystem can AI agents provide consistent, reliable, and valuable support.

Mattia A. Ciampa
Product Development
Senior Manager
Accenture

Simulation Generation

Recent research demonstrates the increasing autonomy of multi-agent GenAI frameworks for automating simulation workflows, particularly in CFD simulations. These systems translate natural language inputs into executable simulation setups and optimize simulations parameters, reducing the need for expert intervention.

Chen et al. (2024) and Dong et al. (2025) both present multi-agent systems capable of end-to-end CFD simulation based solely on natural language. While Chen et al. (2024) present MetaOpenFOAM, a system that employs a RAG approach using CFD tutorials, NL2FOAM by Dong et al. (2025) replaces RAG with fine-tuning on 28,000+ simulation configurations, enabling more robust domain-specific code generation without external lookups. Both frameworks include agents for requirement interpretation, input file generation, simulation execution, and error handling. Chen et al. (2025a) expand MetaOpenFOAM by OptMetaOpenFOAM, which integrates automated sensitivity analysis and parameter optimization, raising the autonomy and accessibility of simulation optimization for non-experts. Similarly, Pandey et al. (2025) and Yue et al. (2025a) propose multi-agent frameworks that leverage RAG databases to embed domain-specific knowledge from prior setups. These systems refine the roles of agents to include requirement parsing, configuration generation, and iterative correction through error analysis, demonstrating enhanced simulation validity and modeling accuracy. In another publication, the authors automate the approach and extend, among other things, the tool interoperability capabilities of the multi-agent system using MCP (Yue et al. 2025b). Feng et al. (2025) introduce another multi-agent framework that transforms natural language queries into fully automated, reproducible CFD simulations with rigorous reliability standards, demonstrating accessibility and precision across diverse flow problems. The framework consists of different agents, which are responsible for pre-processing, prompt generation, simulation execution and post-processing. Due to the increasing popularity of GenAI and agentic AI in CFD simulations, benchmark suites for evaluating the performance of LLMs in CFD workflows have recently been published (Somasekharan et al. 2025).

Beyond CFD, Hou et al. (2025a) explore FEA simulation generation using a GNN to retrieve and adapt similar simulation code segments, enabling LLMs to generate valid FEA input files. In the field of multibody dynamics, Möltner et al. (2025) show the feasibility of LLM-based simulation generation and evaluation, despite limitations in parameter interpretation.

The highly advanced applications in the field of simulations demonstrate the value that GenAI can offer in this domain. However, several challenges still need to be addressed.

- **Limited generalization**: While surrogate models have demonstrated promising results, their performance often declines when applied to unseen scenarios or highly nonlinear, multi-physics problems. Even PINNs struggle to maintain accuracy when domain knowledge is incomplete or difficult to encode, raising concerns about the robustness and physical plausibility of AI-generated simulation outputs (Herrmann & Kollmannsberger 2024; Sunil & Sills 2024).
- **Insufficient data availability:** Many simulation tasks, particularly outside well-researched domains like CFD, lack large, annotated datasets required to train reliable AI models. Although fine-tuning on task-specific configurations, as demonstrated by Dong et al. (2025), offers a solution, the general applicability of such approaches is limited by the cost and effort of curating domain-specific training data at scale.
- **Trust and Interpretability:** Despite advances in automation and multi-agent orchestration (e.g. Chen et al. 2024; Yue et al. 2025a), GenAI frameworks for simulation are often not seamlessly integrated into standard CAE toolchains. Furthermore, the lack of interpretability (Herrmann & Kollmannsberger 2024), transparent error handling, and

self-validation capabilities hinders user trust, especially in safety-critical or regulatory contexts, where engineers must retain oversight and accountability for simulation outcomes (Möltner et al. 2025).

The maturity level of simulation applications in the literature can already be considered high and, by overcoming the identified challenges, can potentially even be elevated according to the automation levels shown in Figure 20.

Level 0 **Manual Simulation Engineer**	**Level 1** **AI-Assisted Simulator**	**Level 2** **AI-Simulation Decision Support**	**Level 3** **AI-Simulation Optimizer**	**Level 4** **AI-Simulation Generator**	**Level 5** **Autonomous Simulation Agent**
No AI involvement	AI assists specific subtasks	AI supports specific decision making	AI automates development subtasks	AI takes over complete domain tasks	AI carries out domain processes autonomously
Engineers manually set up, run, and interpret simulations	AI suggests simulation parameters and settings.	AI predicts simulation results based on surrogate models.	AI automates full simulation pre- & post-processing.	AI executes full simulation workflows for defined components.	AI controls the full simulation lifecycle incl. design variation and optimization.

Figure 20: Vertical automation levels in AI-based simulation applications

System Testing

System testing is a critical phase in product development, encompassing both software and hardware verification to ensure functional correctness, reliability, and performance under real-world conditions. In practice, this involves complex and often time-consuming tasks such as generating test cases, executing tests across diverse configurations, analyzing large volumes of output data, and diagnosing the root causes of observed failures.

GenAI and related AI technologies are beginning to reshape system testing by automating these traditionally manual tasks, improving test coverage, accelerating feedback cycles, and uncovering system-level issues earlier in the development process. For software testing, GenAI models can interpret requirements or source code to automatically generate test cases, identify logic flaws, and assist in debugging by tracing error propagation or suggesting fixes (Wang et al. 2024a). In hardware-in-the-loop (HIL) or test rig environments, AI supports real-time signal analysis, anomaly detection, and pattern recognition, reducing the engineering effort required to interpret high-frequency, high-volume sensor data.

According to recent developments, the application of GenAI in system testing can be broadly categorized into three major task domains:

- **Test Generation**, where AI algorithms generate and optimize test cases from requirements, specifications, or source code
- **Test Debugging**, in which GenAI supports fault localization, failure prediction, and code-level issue resolution
- **Test Data Analysis**, especially in hardware testing, where GenAI enables intelligent processing of sensor data, identification of failure patterns, and extraction of insights from large-scale test logs or rig outputs.

Test Generation

Test case automation plays a pivotal role in increasing the efficiency, coverage, and consistency of system testing, particularly in complex software systems. Traditionally reliant on manually written test cases, recent advances in GenAI have introduced new ways to automatically generate, select, and optimize test cases based on natural language requirements, source code, or behavioural properties. These techniques not only reduce engineering effort but also improve test relevance, coverage and adaptivity in evolving development environments.

Recent research highlights the increasing potential of GenAI to automate and enhance test case generation across software and cyber-physical systems. Birchler et al. (2023) propose a machine learning-based method to selectively skip test cases unlikely to uncover faults in self-driving vehicle software, significantly improving the cost-efficiency of large-scale test campaigns. Etemadi et al. (2025) introduce CHECKPROP, a novel LLM-based approach for generating property-based tests that verify system behaviour over a wide range of inputs, supporting both design-time verification and runtime monitoring in cyber-physical environments. A growing trend is the integration of GenAI-driven testing with broader development workflows. Huang et al. (2023) present a multi-agent architecture where test case generation, execution, and feedback are coupled with automated code refinement through a programmer agent. This aligns with findings from Jin et al. (2024), who observe a shift toward LLM-based agents that interconnect test case creation, debugging, and software improvement, paving the way for more autonomous, adaptive testing systems. Amyan et al. (2024) present an

NLP-driven approach using BERT and Word2Vec that automatically derives and executes fault injection test cases from functional safety requirements on HIL platforms, achieving over 91% accuracy and significantly improving the efficiency of ISO 26262–compliant automotive safety validation. An LLM-based method to automatically generate PLC test cases from function block code is proposed by Koziolek et al. (2024), showing that it is fast and effective for low-to-medium complexity programs. Wynn-Williams et al. (2025) demonstrate that LLMs can translate informal automotive test specifications into executable test scripts with reasonable accuracy, while emphasizing the importance of prompt design, model choice, and retrieval mechanisms for industrial applicability. Milchevski et al. (2025) propose an AI-powered assistant that leverages LLMs and structured intermediate representations to generate system-level test specifications, reducing development effort by 30–40% and improving accuracy and reliability in safety-critical domains. Ye et al. (2025) introduce UVM2, an LLM-powered verification framework that automates UVM testbench generation and refinement, achieving up to 38× faster setup and outperforming state-of-the-art solutions in code and function coverage for industrial-scale hardware designs.

Industry Insights into Test Case Generation

In the context of embedded software development, the need to accelerate the Test & Validation phase has become increasingly critical. Traditionally, test automation required frequent manual updates whenever software requirements changed, leading to delays and inconsistencies.

To address this, an Agentic AI approach was introduced, leveraging state-of-the-art LLMs and Retrieval-Augmented Generation. This setup enabled the automated generation of test cases directly from evolving requirements and validation results. Implemented via a generative AI platform with agentic workflows, the system continuously adapts test automation scripts, ensuring alignment with the latest development inputs. The result was a noticeable increase in efficiency and quality, even though formal metrics were not captured.

A key insight from this initiative was the importance of integrating Agentic AI with a robust knowledge graph or RAG system. This combination proved effective only when deployed within a broader GenAI infrastructure, where agents and humans collaboratively maintain and update data. This ensures the AI can deliver meaningful and reliable automation support throughout the V-Model development cycle.

Marcus Hammes
Technical Director
Accenture

Test Debugging

GenAI is increasingly being applied to streamline debugging workflows by detecting software defects, suggesting fixes, and improving the overall interaction between developers and automated tools. These AI-assisted debugging approaches aim to reduce the time and effort required to identify and resolve issues, while also enhancing developer trust and transparency.

Wang et al. (2025a) introduce *Copilot for Testing*, an integrated debugging and testing system embedded directly within the software development environment. It continuously monitors codebase changes to detect bugs, generate relevant test cases, and propose fix suggestions in real time. This tight integration accelerates the feedback loop between coding and testing, enabling faster, more iterative development cycles. Focusing on developer interaction, Kang et al. (2025) propose an LLM-based debugging framework that not only identifies and resolves code issues but also explains its reasoning process to developers. This added transparency fosters greater trust in AI-driven debugging and improves user acceptance in professional development environments. To support rigorous evaluation of such systems, Tian et al. (2024) present a benchmarking framework for LLM-based debugging tools, offering standardized

scenarios and metrics to assess the effectiveness and reliability of AI-generated fixes and diagnostics. Yao et al. (2024) propose HDLdebugger, a RAG-based and fine-tuned LLM framework that automates debugging of Hardware Description Language code, outperforming 13 baselines and achieving up to 81.93% pass-rate in chip design tasks.

Test Data Analysis

Compared to software testing, hardware testing for complex mechatronic systems presents greater challenges due to its iterative nature and the layered integration of model-in-the-loop, software-in-the-loop, and HIL simulations (Sadri-Moshkenani et al. 2022). The heterogeneity and physical dependencies of these processes make it difficult to apply systematic AI testing strategies. As a result, GenAI applications in this context focus primarily on analyzing sensor signals and test data collected from physical test rigs.

Chen et al. (2025b) introduce FaultGPT, a system that leverages GenAI to generate automated fault diagnosis reports directly from vibration signals, offering fast and interpretable feedback in hardware testing environments. Similarly, Alsaif et al. (2024) present a multimodal LLM fine-tuned for fault detection and diagnosis in Industry 4.0 scenarios. Their approach processes diverse data types, such as images, audio, vibration signals, video, and text, to provide comprehensive diagnostics and actionable guidance to test engineers. Compared to conventional ML techniques, these multimodal models can dynamically support users throughout the testing process. Abboush et al. (2024) propose a novel framework that uses automated fault injection and HIL simulation to generate high-quality, representative real-time datasets for AI–assisted validation of automotive software systems. In another publication, the authors propose an ML-assisted failure analysis approach that employs LSTM models to automatically detect and classify known and unknown faults for the real-time validation of automotive software systems (Abboush et al. 2025). Additionally, Auer et al. (2025) propose generalizable time-series models for anomaly detection and forecasting that can be applied out of the box without fine-tuning. These models offer scalable solutions for real-time signal analysis across different hardware setups and use cases. Presentedj developments highlight how AI enhances data interpretation and decision-making in HIL testing by transforming complex sensor data into valuable diagnostic insights and recommendations.

Further publications deal with LLM-assisted log parsing of diverse HiL documents. For example, Xiao et al. (2024) propose LogBatcher, a training-free LLM-based log parser that clusters and batches logs to reduce overhead, achieving efficient and cost-effective log parsing across diverse datasets. Similar approaches called LogParser-LLM and LLMParser are presented by Zhong et al. (2024) and Ma et al. (2024b), respectively. An extensive survey on the use of LLMs for event log analysis is provided by Akhtar et al. (2025).

The highly repetitive nature of testing tasks offers significant potential for automation, although several challenges still need to be overcome:

- **Limited test diversity and low coverage:** Generating diverse and comprehensive test inputs remains a challenge for LLMs, as they often struggle to explore the full behavioral space of the software under test. Despite strategies like mutation testing and fuzzing, current approaches still result in low line and branch coverage, limiting the effectiveness of automated testing (Wang et al. 2024a).
- **Challenges in real-world application:** Applying LLMs in industrial software testing faces practical barriers, including concerns about data privacy, limited computational resources, and the need for organization-specific fine-tuning. Many companies opt for

open-source models, which often underperform without high-quality, domain-specific training data, which poses a significant hurdle for widespread adoption in production environments (Wang et al. 2024a).

- **Complexity and heterogeneity of hardware testing workflows:** Hardware testing, especially in mechatronic systems, involves layered approaches such as model-in-the-loop, software-in-the-loop, and HIL testing (Sadri-Moshkenani et al. 2022). The diversity of hardware setups, sensor configurations, and test environments makes it difficult to standardize GenAI applications, limiting scalability and requiring extensive adaptation for each use case.

Figure 21 shows the different automation levels in GenAI-based system testing applications.

Level 0 **Manual Test Developer**	**Level 1** **AI-Assisted Test Writer**	**Level 2** **AI-Supported Test Designer**	**Level 3** **Semi-Automated Test Optimizer**	**Level 4** **Task-Autonomous Test Generator**	**Level 5** **Autonomous Testing Agent**
No AI involvement	AI assists specific subtasks	AI supports specific decision making	AI automates development subtasks	AI takes over complete domain tasks	AI carries out domain processes autonomously
Engineers handle test and IVV activities fully manual.	AI suggests software tests and analysis scenarios.	AI identifies critical test coverage areas for hardware and software.	AI generates test cases and defines boundary conditions.	AI executes, adapts, and evolves test cases and environments.	AI manages end-to-end test case & script generation and log analysis.

Figure 21: Vertical automation levels in AI-based system testing applications

Release Management

As product complexity grows, particularly in regulated industries such as automotive, aerospace, and medical technology, release management has become a critical pillar of product development. It encompasses not only the coordination of software and hardware release cycles but also the generation of technical documentation and the assurance of regulatory compliance. These tasks are typically labour-intensive, repetitive, and involve navigating large volumes of heterogeneous data across version histories, change requests, test cases, requirements, and configurations.

AI is now being explored as transformative tool to streamline release workflows. AI algorithms can automatically extract, summarize, and synthesize relevant information from technical artifacts to generate documentation, traceability records and compliance reports. By embedding domain-specific language models into development pipelines, organizations can monitor compliance more continuously and reduce the manual overhead of maintaining up-to-date regulatory records. AI systems can also intelligently link distributed and unstructured data to derive insights, generate impact analyses, and trace the evolution of functionality across releases.

AI applications in release management can be broadly categorized into three major task domains:

- **Documentation Generation**, where GenAI is used to generate user manuals, maintenance guides, or product documentation from various inputs such as source code, change logs, and configuration files.
- **Compliance Monitoring**, where AI systems assist in monitoring, extracting, and structuring compliance-related information to support audits and reduce the risk of non-conformity.
- **Release Note Creation**, where GenAI generates clear and consistent release notes by synthesizing information from change requests, commit messages, test results, and requirements, helping to ensure traceability and improve communication across stakeholders and product versions.

Documentation Generation

Product documentation, such as user manuals, maintenance guides, API references, and safety instructions, is essential for ensuring usability, maintainability, and regulatory compliance. Traditionally, documentation is created manually by technical writers or engineers, a process that is often disconnected from fast-paced development cycles. As product complexity increases and regulations evolve, the need for up-to-date, traceable, and standardized documentation translated into different languages has become more pressing. AI offers a promising solution by automatically generating technical documentation from structured and semi-structured data sources such as requirements, source code, system configurations, change logs, and design specifications. This not only reduces manual effort but also enables near real-time updates to documentation as the underlying product evolves due to continuously applying product changes.

Sovrano et al. (2025) demonstrate the use of LLMs for generating software-related technical documentation that complies with the European AI Act. Their approach focuses on aligning AI-generated documentation with regulatory requirements by interpreting legal constraints and

translating them into structured descriptions of system behaviour, data usage, and risk management. This work highlights the potential of GenAI to bridge the gap between legal compliance and technical clarity, particularly in regulated domains where documentation must serve both engineering and auditing purposes. Tao et al. (2024) propose LLM-R, a framework for the generation of maintenance schemes based on hierarchical agents and RAG, which is intended to enhance the equipment operation efficiency in different industries, such as aviation, energy and transportation. Shi et al. (2025) present a method based on KGs, RAG and Chain-of-thought prompt engineering for the generation of accurate, structured maintenance guidance documents. They demonstrate that they significantly improve content precision and structural controllability compared to prompt-only approaches. Khoee et al. (2024) introduce GoNoGo, an LLM-based multi-agent system that streamlines automotive software release decisions by automating data analysis and supporting risk-sensitive deployment choices, thereby reducing manual intervention and accelerating release processes.

Industry Insights into Technical Documentation

Agentic AI is redefining the landscape of technical publications by automating repetitive authoring tasks, accelerating content delivery by up to 30%, and reducing manual effort to achieve ~20% cost savings. Beyond efficiency, validation agents significantly improve accuracy by ensuring compliance with regulatory standards and minimizing human error. Multi-agent systems also enable seamless collaboration across authoring disciplines - writing, editing, illustration, and program management - creating more consistent and integrated outputs.

The technical foundation relies on a modular agent architecture that scales across diverse products and geographies. Generative agents powered by LLMs handle content drafting and refinement. Retrieval agents integrate vector databases with hybrid keyword and semantic filters to ground content in authoritative sources. Validation agents leverage NLP models and rule-based engines to automate compliance checks, while personalization agents use embeddings and recommendation models to adapt documentation to specific user contexts. These agents are orchestrated via event-driven workflows, with APIs connecting them to product data repositories, compliance systems, and feedback loops. A shared memory layer spanning short-term context and long-term knowledge can ensure continuity and adaptability across the authoring process.

As adoption matures, Agentic AI will evolve into self-optimizing systems, fostering innovation, accountability, and higher-value technical publications.

Ronobijay Bhaumik
Practice Leader
Accenture

Ashish Wadjikar
Associate Director
Accenture

Compliance Monitoring

Ensuring compliance with industry-specific regulations and legal frameworks is a central aspect of product release management. Compliance tasks often involve interpreting complex legal texts, mapping requirements to technical artifacts, and generating documentation for internal reviews or external audits. Traditionally handled through manual processes, compliance checks are time-consuming, prone to oversight, and difficult to scale as products and regulations evolve.

Hassani (2024) presents an LLM-supported framework for regulatory analysis that assists engineers and legal experts in identifying relevant legal clauses and aligning them with product documentation. Using RAG-techniques, the system enables semi-automated compliance reporting by highlighting potential gaps without replacing human judgement. This supports compliance-by-design approaches while reducing manual effort in interpreting regulatory texts. Han et al. (2025) propose a modular RAG-based system that automatically determines the applicability of medical device standards across jurisdictions, achieving interpretable, traceable justifications and significantly improving compliance reasoning compared to

retrieval-only or rule-based approaches. Arora et al. (2024b) introduce CompliAT, a framework that leverages LLMs to ensure terminology consistency, classify assistive technology products, and trace specifications to regulatory requirements, thereby improving compliance, accessibility, and safety in release management and technical documentation. Madireddy et al. (2025) develop an LLM-driven framework that semi-automates building code compliance checking by translating regulatory requirements into executable scripts, thereby reducing manual effort and improving both accuracy and efficiency in regulatory verification for construction projects.

Release Note Creation

Release notes play a critical role in communicating changes, improvements, and known issues to stakeholders at each product version milestone. Traditionally compiled by release managers, this task requires the aggregation of inputs from multiple sources, including test reports, issue trackers, and version control systems, making it highly dependent on manual expertise. GenAI presents a promising solution to streamline and automate this process. By synthesizing product data from across the development pipeline, LLMs can generate coherent, audience-tailored release notes that improve traceability and decision-making.

Daneshyan et al. (2025) demonstrate an LLM-based pipeline for automated release note creation tailored to project-specific domains, significantly reducing manual workload and ensuring consistency across versions. Similarly, Wu et al. (2024b) introduce a co-pilot system that assists release managers not only by summarizing technical artifacts, such as test outcomes, defect statistics, and code quality metrics, but also by answering strategic queries like “Are we ready to release?” or “What are the open risks?”. In support of the documentation quality, Kumar et al. (2024) propose using LLMs to evaluate the clarity and completeness of bug reports and software artifacts, further enabling high-quality, automatically generated release documentation.

GenAI applications in release management offer huge potential, but key challenges remain critical:

- **Fragmented and unstructured data sources:** Release documentation and compliance reports must synthesize information from diverse sources across all disciplines of the V-model such as change logs, test reports, code repositories and requirement databases. The lack of standardized formats and semantic consistency across these artifacts complicates automated data extraction and summarization by GenAI systems.
- **Context-aware documentation generation:** Automatically generating accurate and audience-specific release notes or compliance reports requires a deep understanding of the domain, product context, and stakeholder needs. While AI models can generate fluent text, maintaining factual correctness, traceability, and relevance to regulatory standards remains a key challenge. Providing product- and company specific context to the algorithms is a key challenge for GenAI applications in release management but can be tackled through the application of RAG and KGs.
- **Limited trust and validation mechanisms:** GenAI-generated outputs, such as release summaries or compliance insights, require validation for correctness and completeness. However, robust quality assessment frameworks for AI-generated documentation are still underdeveloped, posing risks to trust and adoption in high-stakes industrial release processes, such that manual involvement is still highly required for critical release and compliance documentations.

Figure 22 provides an overview of the automation levels in AI-based release management applications.

Level 0 Manual Technical Writer	Level 1 AI-Documentation Assistant	Level 2 AI-Documentation Planner	Level 3 AI-Documentation Generator	Level 4 Task-Autonomous Documentation	Level 5 Autonomous Documentation Agent
No AI involvement	AI assists specific subtasks	AI supports specific decision making	AI automates development subtasks	AI takes over complete domain tasks	AI carries out domain processes autonomously
Engineers and tech writers manually draft and maintain all documentation.	AI suggests text snippets or formatting for documentation.	AI recommends documentation structure and additional content.	AI automatically generates chapter drafts.	AI takes over specific E2E documentation tasks.	AI controls end-to-end documentation & compliance pipeline.

Figure 22: Automation levels in GenAI-based release management applications

Cross-Domain Applications

While many AI applications in engineering focus on a single engineering domain and discipline, an increasing share of publications exploit the interconnections between domains. Modern engineering processes are inherently cross-functional, with artifacts such as requirements, architecture models, source code, CAD design, test cases, test logs, and release documentation continuously influencing one another. This creates opportunities for AI systems to add value by integrating heterogeneous data sources, ensuring consistency across domains and artifacts, and enabling knowledge transfer beyond domain and disciplinary boundaries.

Such cross-domain applications are especially powerful where engineering complexity, regulatory pressure, and the demand for rapid product cycles converge. By bridging silos, AI does not only enhance efficiency but also reduces errors and strengthens compliance in scenarios where manual synchronization would be error-prone and resource-intensive.

Cross-domain applications can be categorized into

- **Change & Configuration Management**, where AI supports the analysis of change requests, configuration baselines, and related artifacts across domains to ensure consistency, assess impacts, and improve traceability throughout the product lifecycle.
- **Portfolio & Variant Management,** where AI enables the analysis and optimization of complex product portfolios and variant structures by identifying redundancies, predicting market and cost impacts of portfolio decisions, and supporting automated variant derivation and configuration based on technical and business constraints.
- **Program & Project Management,** where AI assists in planning, monitoring, and controlling engineering programs by predicting schedule deviations, cost overruns, and resource conflicts, while also automating reporting and decision support through intelligent analysis of project data and dependencies.
- **Cross-Domain Multi-Agent Applications**, where multi-agent systems carry out cross-domain development tasks autonomously, modify artifacts, implement changes, and automate cross-domain engineering processes.
- **Cross-Domain Context Management**, where AI extracts and links knowledge from heterogeneous engineering domains and stores it in vector (RAG) or graph databases (KG) to enable efficient retrieval and reasoning for downstream AI applications.

Configuration and Change Management

AI can significantly enhance configuration and change management processes by enabling faster identification of dependencies, predicting potential impacts, and supporting decision-making in complex development environments. AI-based methods have already been widely deployed within software impact analysis (Samhan et al. 2024) and show strong potential to support the efficient development of complex physical products as well (Burggräf et al. 2024). This helps companies reduce risks, improve consistency, and accelerate change implementation.

In automotive software development, El Asad et al. (2025) propose a RAG-assisted LLM concept, that predicts impacts caused by software updates in vehicle manufacturing and enables earlier detection of risks that might otherwise appear only at later stages. Treshcheva et al. (2025) create traceability links between requirements and test scripts, which are essential for performing change management activities such as impact analysis. Zhang et al. (2025e) introduce MBSE 2.0, a next-generation systems engineering framework that integrates AI, model governance, and cross-domain collaboration to overcome the limitations of traditional MBSE. The authors state that AI enhances enterprise change management by moving beyond

static, manually defined traceability toward intelligent association, where technologies like KGs and LLMs automatically infer and update relationships between requirements, architectures, and simulations.

Industry Insights into Change & Config. Mgmt.

Managing product configurations and technical changes has become increasingly complex. Every modification must be evaluated against a large set of interdependent requirements, design elements, test results, and sourcing constraints.

Traditionally, engineers spend substantial time searching across disconnected systems and documents to trace the impact of a change, often leading to delays or incomplete assessments. Artificial Intelligence offers a new approach by combining Knowledge Graphs with Large Language Models (LLM). Combined, they create a structured map of product data, linking information from heterogeneous sources into a consistent and navigable network.

On top of this, LLMs can retrieve and interpret the relevant context, guiding engineers quickly to the connections that matter most. This makes it possible to understand the implications of an Engineering Change Request in minutes rather than days, while maintaining end-to-end traceability. The benefits are more reliable decisions, greater transparency in change processes, and improved collaboration between disciplines throughout the product lifecycle. These concepts provide a promising path to finally connecting information silos and realizing true end-to-end, collaborative engineering.

Dr. Jan-Ivo Springborn
Director Config. Mgmt.
Accenture

Florian Böhme
Manager Config. Mgmt.
Accenture

Portfolio & Variant Management

Managing product portfolios and variants, as well as selecting R&D projects, presents significant complexity for companies, requiring advanced methods to ensure efficiency and strategic alignment. AI-driven analytics can help identify market trends, optimize resource allocation, and evaluate trade-offs between cost, risk, and innovation potential. By leveraging predictive modeling and scenario-based simulations, organizations can make more informed, data-driven decisions that enhance competitiveness and reduce time-to-market.

Mehlstäubl et al. (2022) address the challenge of predicting product attribute values for multi-variant product portfolios, where companies offer an almost infinite number of product variants and attribute values must be determined even for previously unbuilt configurations. Their methodical approach utilizes ML to predict product attributes based on customer feature configurations, demonstrating that ML reduces effort and provides more accurate and faster predictions compared to traditional rule-based expert systems. Nielsen et al. (2024) focus on the strategic selection of industrial R&D projects, a complex task influenced by innovation unpredictability, competition, and technological changes. They propose a multi-objective optimization program as a conceptual quantitative framework to systematically analyze R&D projects and optimize corporate objectives by considering project values and risks in a multi-project context. A case study in the renewable energy sector demonstrates how this framework provides optimal trade-offs between portfolio value and risk, enhancing transparency in decision-making.

Program & Project Management

Managing programs and projects in the R&D of complex mechatronic systems places high demands on planning accuracy, cross-domain coordination, and risk management. AI applications can support this process by providing predictive insights into project schedules, resource utilization, and potential bottlenecks across mechanical, electrical/electronic, and software development streams. Through advanced analytics and intelligent decision support

systems, organizations can enhance transparency, mitigate risks early, and improve overall program efficiency and delivery reliability.

Geyer et al. (2025) present an industry case study investigating the role of LLMs in evaluating the quality of epics, which are critical artifacts for communicating software requirements in agile software development. Their user study with product managers indicated high levels of satisfaction, suggesting that LLM evaluations are a viable application for improving epic quality and consistency, while also outlining challenges such as the need for flexibility and domain knowledge. Kumar et al. (2025) introduce a "synthetic teammate" framework to strategically integrate GenAI into product development activities, aiming to enrich and accelerate the overall process. This approach advocates a "human-first" methodology, positioning GenAI as a managed team member that enhances human thinking across problem and customer identification, ideation, concept development, and commercialization, with humans retaining ultimate control and decision-making responsibility.

Cross-Domain Multi-Agent Applications

A key enabler of cross-domain AI applications in engineering is process automation driven by multi-agent systems. Unlike traditional automation approaches confined to single tools or domains, multi-agent systems provide a distributed intelligence layer that can coordinate and execute modifications across heterogeneous engineering environments. Each agent can specialize for a specific task while collectively working toward a common engineering objective. By communicating and negotiating with one another, these agents enable consistent propagation of changes, ensure alignment of artifacts across domains, and reduce the manual effort typically required to synchronize complex toolchains.

Wang et al. (2025b) present a multi-agent framework for autonomous mechatronics design including four agents being responsible for mechanical, electronic, control and software engineering, respectively. Validated through the development of an autonomous water-quality monitoring vessel, their agentic framework demonstrates how cross-disciplinary agents combined with structured human feedback can lower expertise barriers and enable scalable, real-world engineering innovation. A similar approach is performed by Elrefaie et al. (2025), who choose a multi-agent framework consisting of CAD, styling, simulation and meshing agent to generate 2D automotive concepts, transform it into a 3D CAD model and run aerodynamic simulations for generated 3D models. Orchestration between agents can accelerate the iterative design process while satisfying industry-standard engineering constraints. Jin et al. (2025) propose a two-stage multi-agent framework that integrates generative design agents with a surrogate-based drag prediction agent, enabling the automated transformation of ambiguous requirements into validated 3D automotive concepts while balancing aesthetics and aerodynamic performance. Ocker et al. (2025) introduce a vision language model based multi-agent architecture for CAD that mirrors industrial development teams by combining requirements engineering, CAD code generation, and vision-based quality assurance, enabling iterative, user-in-the-loop creation of parametric models from sketches or textual descriptions. Ni et al. (2025b) present CADDesigner, an LLM-powered agent that generates high-quality CAD modeling code from textual requirement descriptions and sketches using a novel context-independent imperative paradigm, enhanced by iterative visual feedback and a knowledge base for continuous improvement.

Recent research published in 2025 shows a clear shift toward cross-domain multi-agent applications in engineering and design. Multi-agent systems are increasingly being used to connect tasks such as requirements analysis, design generation, simulation, and validation across different domains. This trend highlights a growing focus on orchestrated, collaborative AI agents, and it is expected that the field will gain strong traction in the coming years.

Context Management

Cross-domain context management in engineering has gained increasing importance as product development processes continue to grow in complexity. With rising system integration, the need for centralized context management, data interconnectivity, and the systematic identification of cause-effect relationships has become even more critical. In practice, knowledge about impact chains and interdependencies between domains and disciplines is scattered across distributed development teams. Consequently, alignment to the impact of changes is often time-consuming and significantly slows down the overall product development process.

Recently, advances in AI research have introduced methods that address these challenges (see also Section 04 Context Management). Techniques such as RAG and KGs allow for scalable and automatable knowledge extraction and retrieval, thus enabling cross-domain and cross-disciplinary context management. These methods connect information and artifacts from different domains, providing engineers with faster access to relevant insights and improving decision-making in complex development environments. The publications discussed in the following passage illustrate how such approaches leverage AI-driven methods to interlink heterogeneous development artifacts, ultimately accelerating knowledge access across domain boundaries.

Tong et al. (2024) propose a knowledge recommendation framework for PLM data based on KG and GNN. Their approach provides PLM users with accurate, context-aware knowledge access across product domains and product views (conceptual, design, manufacturing, purchasing, sales, aftermarket) improving efficiency in design and data modeling. Kasper et al. (2024) introduces a graph-based data model of the digital thread that interconnects product lifecycle phases, data models, processes, and IT systems. The authors focus on the define, design, produce, and operate phases, demonstrating that graph databases offer superior performance for recursive operations on networked data compared to relational approaches, and highlight future research needs in integrating dynamic processes such as quality and change management. Ryś et al. (2024) propose a framework based on KGs and ontologies that captures workflow concepts, modeling artefacts, and their interrelations, providing the foundation for establishing traceability across artifacts as well as enabling knowledge retrieval and reuse. The authors validate their approach using both a simple spring–mass–damper example and a real-world engineering scenario involving a drivetrain smart sensor system, demonstrating its applicability and benefits such as improved artifact management, reduced information retrieval time, and enhanced cross-domain reasoning. Darm et al. (2025) propose an LLM-based approach for the automated verification of requirement fulfillment. In their study, requirements are represented as graph structures, and an LLM is employed to reason over these graphs. Using two early-stage Capella SysML models of space missions with associated requirements as examples, the model can determine whether specific requirements are satisfied by analyzing the structural and relational information encoded in the graphs. A cognitive digital thread tool chain to improve model versioning in MBSE is presented by Wu et al. (2025). The tool chain supports conflict detection and resolution across diverse modelling languages and KGs generated during versioning provide reasoning capabilities that enhance traceability and decision support. The approach is validated using the example of a landing gear system, demonstrating higher efficiency than conventional model versioning.

Jiang et al. (2025) present a two-stage RAG framework that integrates design principles and sustainability strategies to provide contextually relevant, early-stage guidance for sustainable product development, significantly improving design outcomes and supporting the transition to a circular economy. Xiong et al. (2025) propose Domain-Rule-based RAG, a framework that combines domain-specific KGs, rule-based reasoning, and digital twin technology to enhance knowledge-driven aircraft design. By dynamically constructing KGs with a hybrid R2D-LLM

approach and integrating rule-based retrieval into a RAG pipeline, DR-RAG improves retrieval accuracy, decision transparency, and design efficiency in complex engineering contexts.

Recent advances such as RAG, KGs and GraphRAG show significant potential for automating context management and enabling the realization of the digital thread and comprehensive, cross-domain traceability. First applications can already be found in literature, demonstrating their value for linking artifacts and improving decision-making and accelerated knowledge reuse in engineering. These technologies are key to connecting domains and disciplines, addressing one of the central challenges of modern product development. We therefore expect a strong increase in research efforts and integration within engineering tools in this field in the coming years.

Use Case Summary

In the use case Section, we present a variety of AI use case classes mapped to the six core disciplines of the V-model in product development. These use cases highlight the broad applicability of GenAI and LLMs in engineering workflows and are summarized in Figure 23.

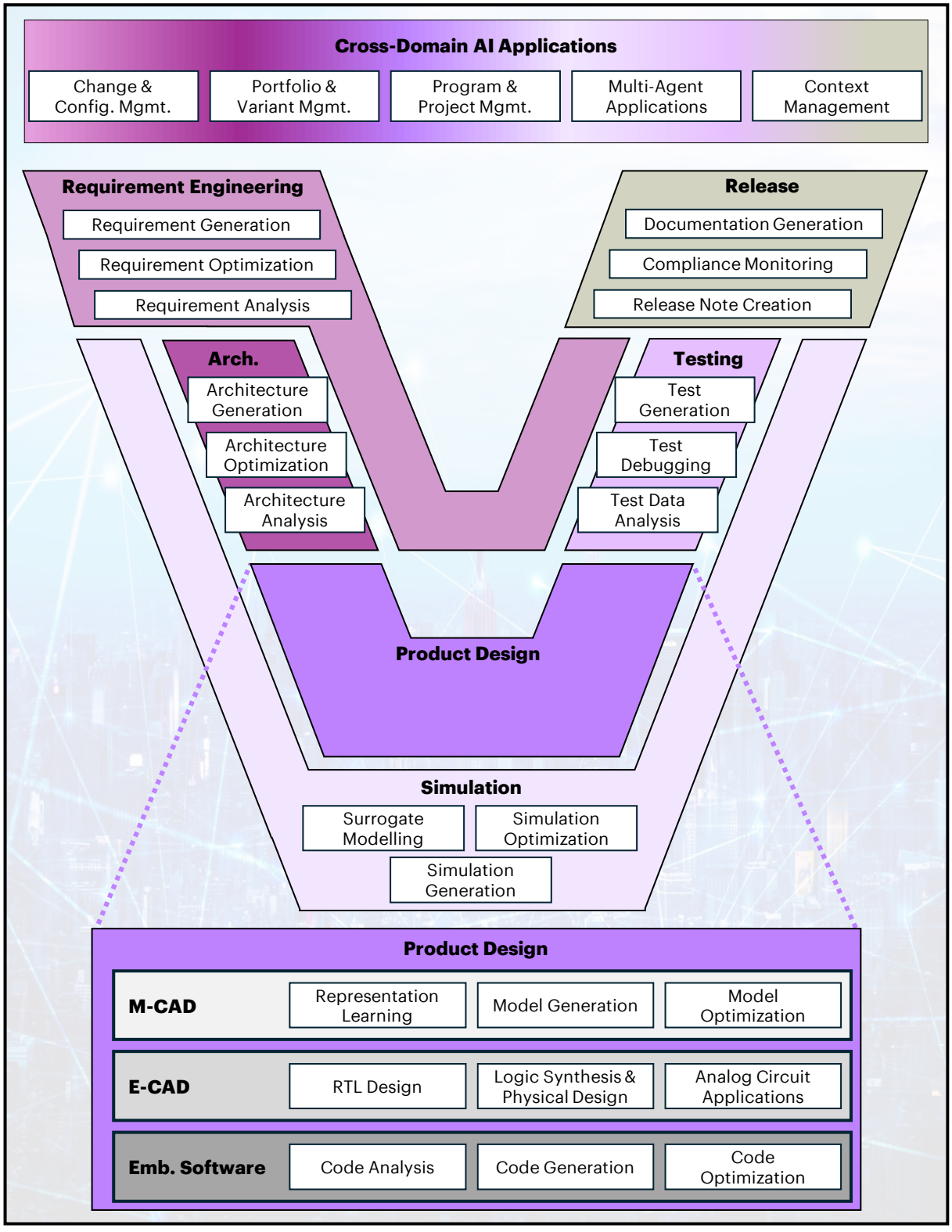


Figure 23: Overview over all use cases across the V-model's development domains

Most identified applications focus on the generation and optimization of engineering artifacts, leveraging existing development outputs to synthesize new content. These generative approaches aim to accelerate and enhance tasks such as requirements formulation, architectural design, product modeling, simulation, testing, and documentation. Creation of traceability between development artifacts is another emerging area of research, though current implementations are fragmented and typically confined to individual development disciplines (Fuchß et al. 2025a, Hassine 2024). A holistic, cross-disciplinary traceability solution with development artifacts from all engineering disciplines of the V-model has not yet been realized.

Technically, the use cases rely either on RAG-based knowledge systems or, in more advanced cases, on fine-tuned LLMs tailored to discipline-specific data. Applying LLMs in the disciplines captured in the upper stages of the V-model, such as requirements engineering, architecture design, testing, and release documentation, is generally more straightforward. This is because the artifacts in these disciplines are primarily text- or code-based, making them well-suited to the strengths of current LLM technologies. In contrast, LLM applications in the lower sections, especially in the design and simulation of physical components (e.g., CAD, FEM, CFD), face significantly higher complexity. These domains require the generation of high-resolution, physically plausible 3D data, which remains a considerable challenge for current generative AI models. However, a noteworthy trend is the movement toward the compilability of design (see CADQuery 2024 and Guan et al. 2025) and simulation models (see Pandey et al. 2025 and Yue et al. 2025a), aiming to make these artifacts more accessible to LLM-driven synthesis, optimization and generation.

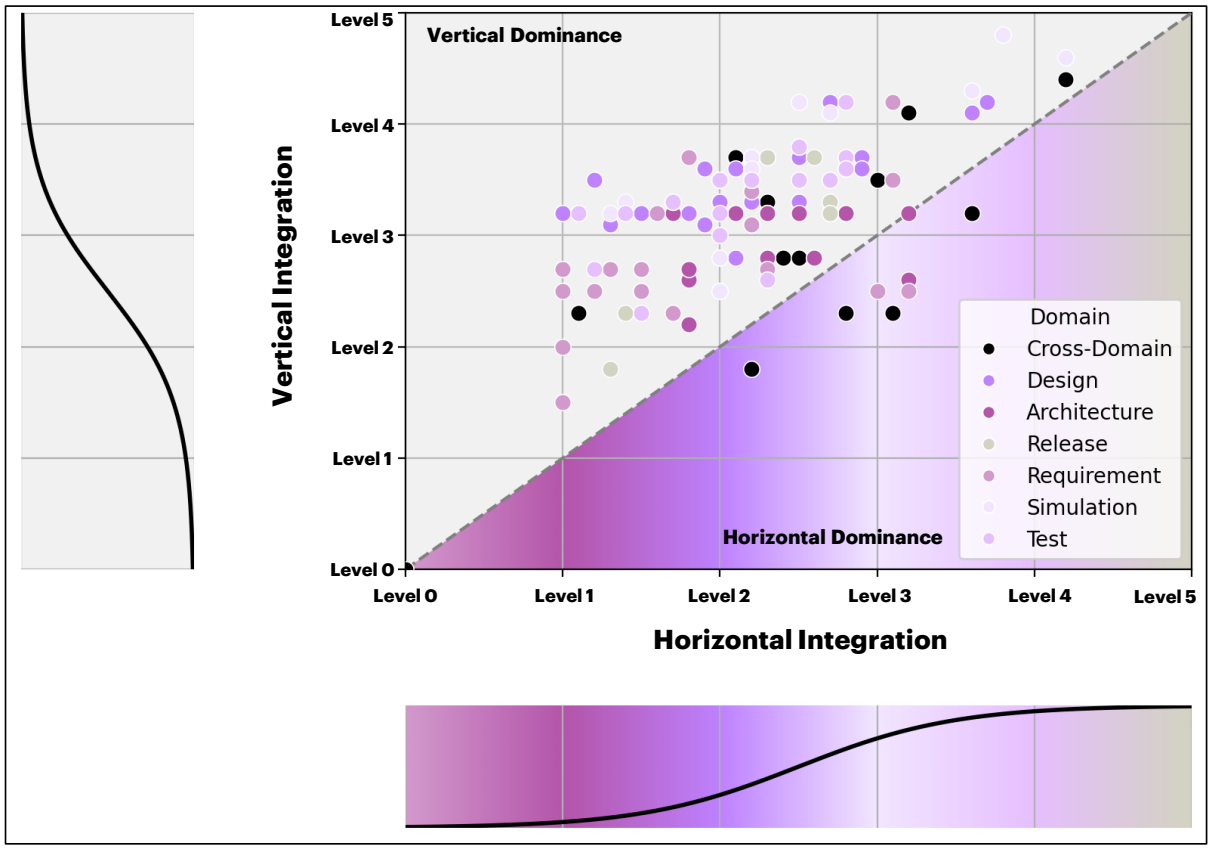


Figure 24: Results from the evaluation of 137 scientific AI publications in engineering with respect to their vertical and horizontal maturity

In the Section *Stages of AI Readiness in Engineering*, we introduced the concept of vertical and horizontal AI integration. Figure 24 shows an evaluation of all 137 publications analyzed in the *Use Case* Section in terms of vertical and horizontal maturity according to the definitions proposed in Figure 15. 129 of the 137 publications have a higher vertical than horizontal maturity level, meaning that their focus is more on solving domain-specific problems than on creating cross-domain links between tools and data. This is a trend we also observe in the implementation of industrial PoCs and use cases. On the one hand, tools and data are highly fragmented in industrial practice, on the other hand, they are typically managed by organizational units that operate within separate areas of responsibility. This means that the AI use case landscape is also oriented toward the fragmentation of tools and data and hierarchical organizational structures. As a result, many AI use cases are being implemented, which ultimately lead to incremental accelerations of domain-specific sub-processes in product development, but do not sufficiently exploit the overarching potential for accelerating overall product development through the implementation of horizontal AI use cases. The realization of the target vision of a digital thread in product development has been discussed for several decades and is considered desirable. With the advent of GenAI and Agentic AI, the incentives for realizing the digital thread are now amplified, as massive reductions in product development cycles are made possible by horizontal AI-fication of engineering.

It is also noteworthy that the current development of GenAI use cases remains largely intra-disciplinary, with efforts primarily concentrated on advancements within distinct engineering fields. Yet, as engineering processes become more connected, the potential for cross-disciplinary use cases across mechanical, E/E and software development will grow. Promising future applications include change and configuration management, cross-domain context management, and integrations with supplier and customer systems (e.g. for cost estimation, demand forecasting and supplier selection use cases), as well as downstream value chain processes such as production planning, M-BOM generation, maintenance, and service.

A closer analysis of the summed up vertical and horizontal maturity levels shown in Figure 24 (i.e., the publications located in the upper right area of Figure 24) reveals a clear trend. The simulation (Yue et al. 2025b, Feng et al. 2025, Chen et al. 2025a, Chen et al. 2024), design (Wu et al. 2024a, Qin et al. 2024), and cross-domain use cases (Jin et al. 2025, Elrefaie et al. 2025) with the highest maturity levels all represent multi-agent systems that automate complex, multi-stage development processes through task distribution, tool interoperability, and agent-based communication. This demonstrates that Agentic AI with its recently acquired capabilities has the potential to significantly increase the degree of automation in development processes.

In the following section, we therefore provide an outlook on multi-agent systems, illustrating how their introduction into engineering affects various dimensions and which measures companies must now implement to fully and rapidly leverage these potentials.

To conclude this chapter, we provide an overview of the AI use cases developed by Accenture and already implemented in industrial environments. Figure 25 summarizes the AI use cases developed and implemented by Accenture in the engineering sector and maps them to the domains of the V-model.

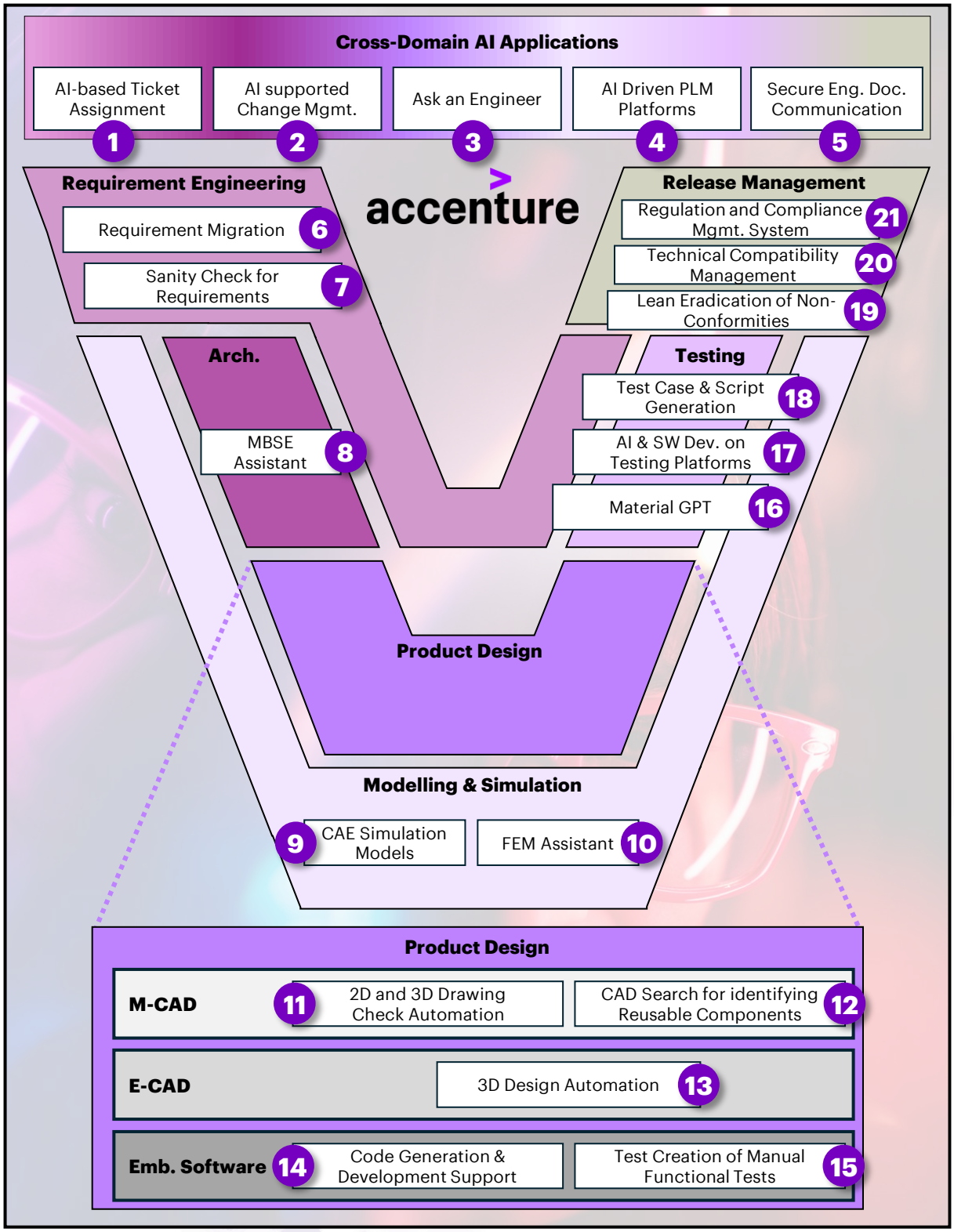


Figure 25: Integration of industrial AI use cases developed by Accenture into the V-model

Outlook

Agentification of Engineering

In the previous Section, we demonstrated that numerous AI use cases are already being employed in literature and industry, but the use case landscape remains fragmented and largely focused on vertical AI integrations. Recent research into GenAI and Agentic AI has massively expanded the portfolio of AI capabilities that have so far been not yet applied in current product development processes. In this Section, we therefore show how fragmented use case landscapes can be overcome by operationalizing these capabilities and providing engineers with AI solutions that ensure a holistic application in the product development process.

As indicated in Figure 2, GenAI and Agentic AI bring new capabilities that are particularly important for engineering of the future. These include

1. **Planning & Reasoning:** capability to answer questions that require complex, multi-step processes with intermediate steps, enabling systematic problem-solving beyond fast, heuristic responses (Li et al. 2025g, Sui et al. 2025)
2. **Orchestration:** capability to automatically coordinate multiple LLM agents and tasks including state tracking, dependency management, independent validation, and compensatory rollback to ensure consistent, reliable execution across distributed workflows (Chang & Geng 2025)
3. **Tool Interoperability:** capability to discover multiple tools, orchestrate their use, and reliably invoke them to enable multi-step workflows across environments (Xu et al. 2025b)
4. **Multi-Agent Collaboration:** capability to enable multiple LLM-based agents to coordinate and manage joint objectives through structured collaboration channels so that they jointly plan, exchange knowledge, and make collective decisions to achieve shared goals (Tran et al. 2025)
5. **Context Management:** capability to preserve essential constraints, state history, dependencies, and reasoning justifications so agents can reliably track, recall, and use context across multi-step workflows, especially when failures or replanning occur (Chang & Geng 2025)
6. **Memory:** capability to store, manage, and retrieve past information including conversation history, task states, and reasoning traces so that agents and multi-agent systems can maintain long-term context, support consistent decision-making, and enable continual learning across interactions (Zhang et al. 2025d)

These capabilities have far-reaching implications for the engineering of the future. Table 2 lists the capabilities and describes their implications for the engineering of the future. It shows that multi-agent systems offer great potential for automation in product development, especially when end-to-end understanding of product architectures, system and process modeling, and access to metadata (stored in graph and vector databases) and engineering data (stored in individual tools and accessible using tool interoperability capabilities) are enabled. A high-level architecture that exploits the potential of the listed capabilities is shown in Figure 26. The capabilities listed in Table 2 are assigned to the system elements.

Table 2: Implications of GenAI and Agentic AI capabilities on engineering

Capability	Implication on engineering
Planning & Reasoning	The development of complex mechatronic products is a highly networked process whose complexity must be mastered through multi-stage decomposition and interface definitions across multiple engineering domains and levels. Leveraging planning and reasoning capabilities can support identifying dependencies and constraints, and dynamically generating and refining product structures, requirements, and design alternatives across the entire lifecycle.
Orchestration	High-level engineering tasks such as implementing product changes involve many steps that must be planned in fine detail and managed adaptively. Breaking down an overall task into subtasks and processing them sequentially while monitoring overall progress and possible dependencies are essential components of engineering activities. With orchestration capabilities, it will be possible in the future to orchestrate complex tasks, continuously monitor their progress, and proactively draw attention to risks.
Interoperability	Engineering toolchains from companies with complex product portfolios often consist of several hundred or even thousands of tools, whose interoperability is ensured by point-to-point integrations or by connection to lifecycle systems. With interoperability capabilities and agentic communication protocols such as MCP, data can continue to be managed in tools and retrieved or modified from the outside. This will ensure compatibility between multi-vendor toolchains and AI applications, while at the same time increasing the requirements for standardized data management.
Multi-Agent Collaboration	As engineering tasks become more complex and orchestration efforts increase, so does the need to deploy multiple interacting agents that work together to achieve an overarching goal. Hierarchical agent architectures consisting of an orchestrator with an end-to-end product view and multiple subagents that communicate bidirectionally with the orchestrator are ideal for this purpose. Together with planning & reasoning, orchestration, and tool interoperability capabilities, entire multi-agent systems for engineering can be designed that automate cross-domain and cross-tool processes, with agents reacting adaptively to states and unplanned events.
Context Management	The provision of context is particularly important for complex product developments, since domain-specific processes, data and syntax play a crucial role. For agents to understand complex systems, architectures, processes, dependencies, and structures, information must be provided centrally and be reflected in graph databases, vector databases or process and system modeling. Access to this data and gaining a higher-level understanding of the overall system structure and processes for developing or modifying the system determines the degree of end-to-end.
Memory	Additional context that is valuable for performing engineering tasks comes from engineers' historical prompt histories, design rationales, and decision logs. By leveraging the memory capabilities of LLMs and Agentic AI, engineering teams can maintain long-term context across development tasks and cycles, ensure consistent and traceable decision-making, and enable continual learning from previous designs and problem-solving episodes. This persistent, structured memory supports complex, multi-stage product development by reducing knowledge loss, accelerating iteration, and improving cross-domain collaboration.

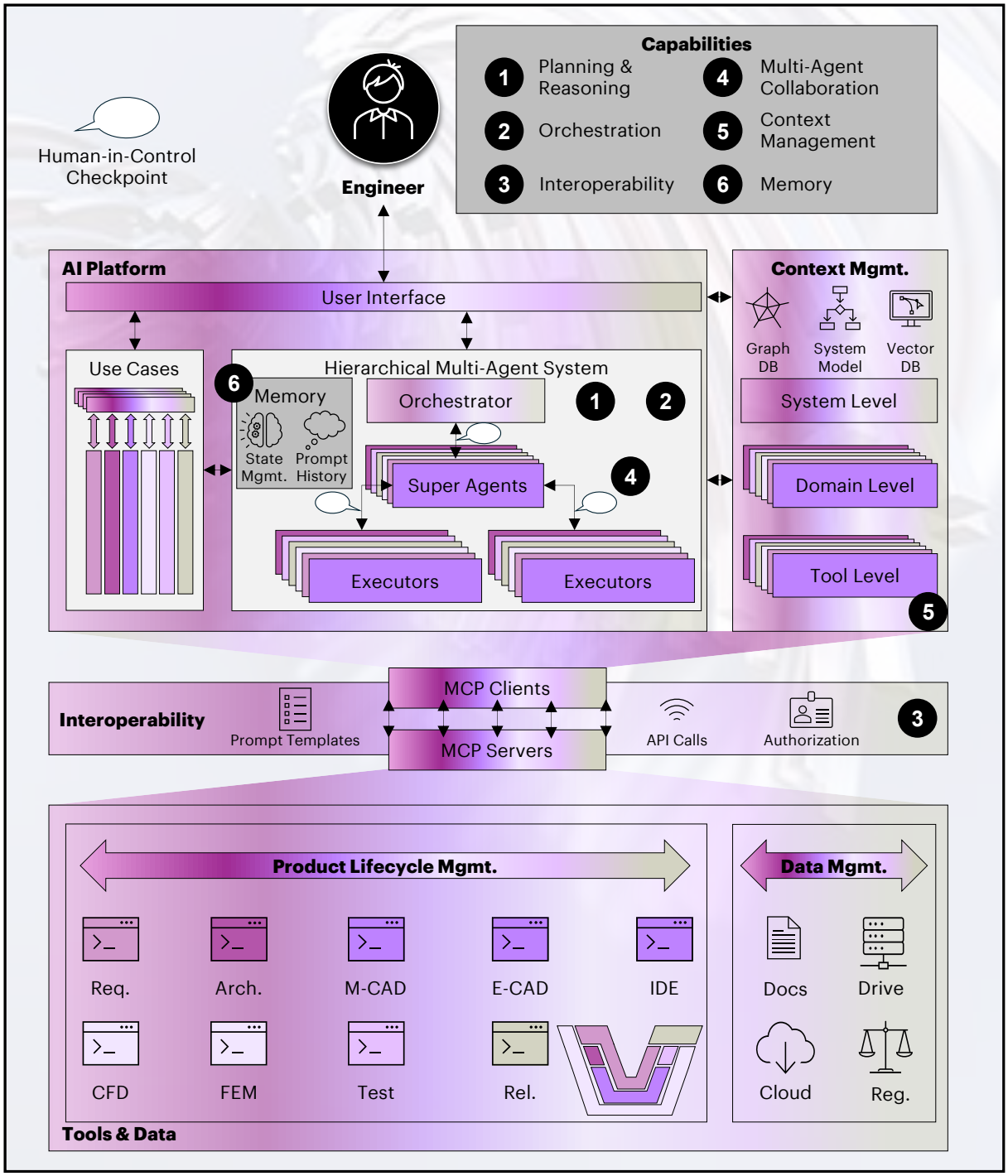


Figure 26: High-level engineering AI architecture including a hierarchical multi-agent system

A user interface provided on the AI platform serves as the interface between engineers and AI applications. This allows access to individual AI use cases as well as a hierarchically structured multi-agent framework that can be used for complex end-to-end engineering tasks. The hierarchical multi-agent system consists of a high-performance orchestrator with reasoning and orchestration capabilities and further agent levels, which are responsible for domain- and tool-specific tasks. In the context management modules, higher-level metadata is systematically managed and made available in graph and vector databases for quick retrieval by agents. Information about product structures, systems and process modeling from MBSE tools is also provided to give agents an overview of the progress of the product development process and complex relationships within the models. Each agent level is linked to a context

module in which data and models are made accessible to the respective agents. The orchestrator requires an end-to-end product view and must therefore access MBSE models and product data that holistically describe the entire product development process. Super agents provide an overview of domain-specific data products and are enriched with domain-specific regulations, product structures, and data architectures. The executors are directly connected to the tools via MCP servers and have tool-specific information in their context modules, such as data structures, formats, and API calls, which they can use to access and modify tool data.

In the automated execution of cross-domain engineering tasks by multi-agent systems, it is essential to introduce Human-in-Control checkpoints. These are designed to review, assess, edit, and ultimately approve intermediate and final results produced by the multi-agent system. It is crucial to ensure traceability, explainability, robustness, and reliability continuously throughout the process. Thus, humans do not remain merely in the loop but in control. This approach requires that employees are proficient in working with AI, able to interpret and validate its results, and thus capable of monitoring compliance and technical feasibility throughout the entire product development process.

Figure 27 shows a simplified example of how multi-agent systems could be embedded in engineering processes in the future.

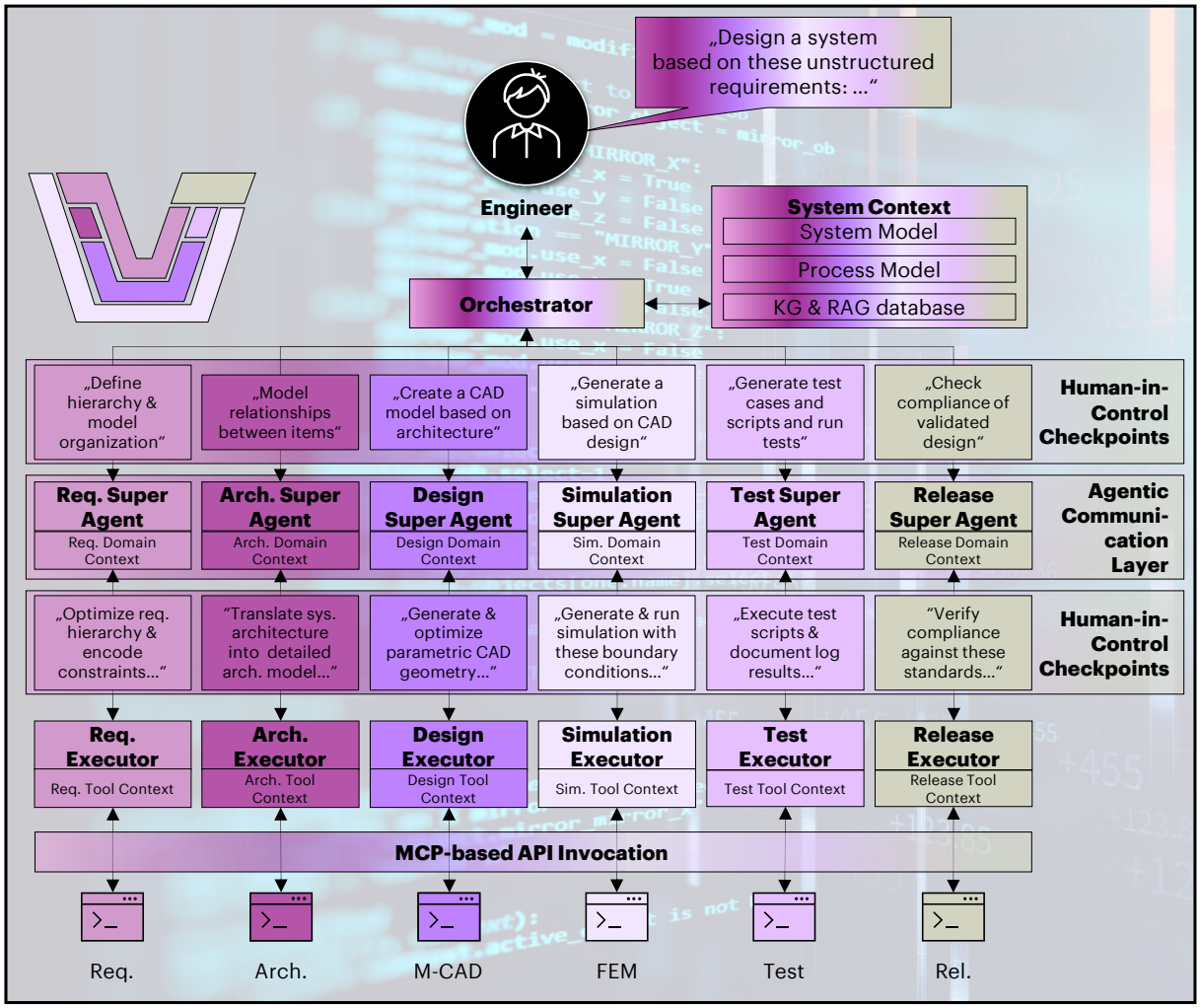


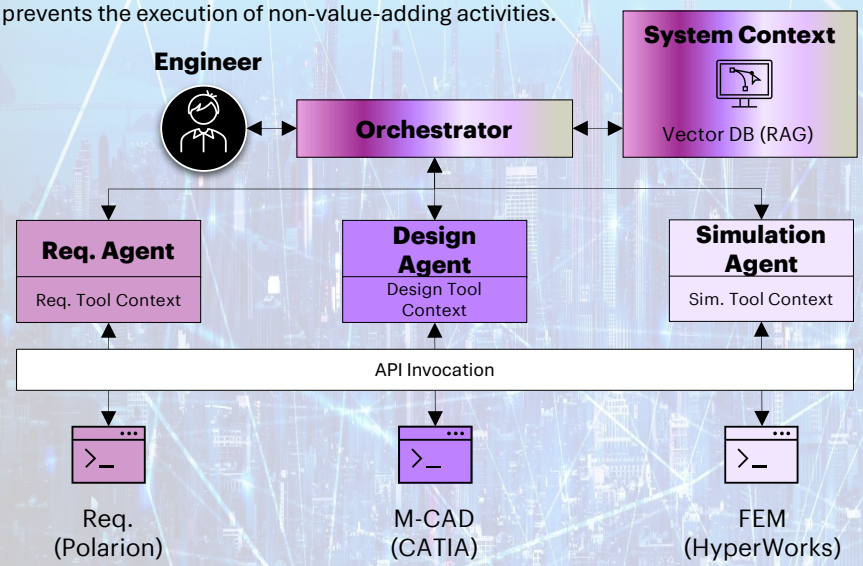
Figure 27: Exemplary and Simplified Multi-Agent Engineering Workflow consisting of a three-level Hierarchical Multi-Agent Architecture, Context Modules, and Tool Interoperability adapted from Larichev et al. (2025)

The engineer sends an abstract prompt to the orchestrator for the development of a technical system, whereupon the orchestrator uses the provided system context to break down the abstract development task and delegate it to domain-specific sub-agents. These agents further enrich the instructions with domain-specific context and then delegate tasks to the executor agents. The executor agents are connected to the appropriate tools via MCP servers, are familiar with the data structures and formats stored in the tools and can perform the development tasks in the tools via API calls. The agents not only communicate top-down, but also inform the corresponding higher-level agent about progress, quality, and the result of the task processing. Bidirectional communication between the agent layers and within one agent layer (see agentic communication layer) is necessary to delegate development tasks, but also to ensure satisfactory task completion and bidirectional information exchange. Between each information transfer across the agent layers, Human-in-Control checkpoints are integrated, allowing humans to influence task execution by providing direct feedback and issuing instructions for rework. Each agent has quality criteria that it checks after the subordinate agent has completed its task and requests rework if necessary. This enables the orchestrator to identify conflicts between the work results of two subordinate agents at an early stage, which in the case of manual product development would only have been noticed in later product development phases of verification and validation, leading to further cost-intensive iterations in the product development process.

Industry Insights into Engineering Multi-Agent Systems

Contemporary engineering environments face increasing product complexity and workforce scarcity, coupled with the demand for accelerated development cycles, strict quality adherence, and reduced manual intervention. To meet these challenges, multi-agent systems offer a scalable and intelligent solution in which AI agents collaborate to automate processes, streamline workflows, and support decision-making throughout the product lifecycle. These agents do not function as isolated tools but interact dynamically. For instance, one agent may manage design modifications in CAD, another may validate simulation models, and a third may extract relevant data from documentation. Together, they establish workflows that shorten iteration cycles and ensure compliance with engineering standards.

To demonstrate this, the Accenture team has developed a multi-agent system capable of integrating with platforms such as Siemens Polarion, 3DS CATIA, and Altair HyperWorks. Each integration features specialized AI capabilities for analysis, design, and simulation. The corresponding agents that control these tools maintain tool-specific context within RAG vector databases, enabling them to access information on engineering task execution via API calls. This coordinated orchestration allows agents to make autonomous decisions and transition seamlessly between tasks and tools. The framework enhances operational efficiency, reduces complexity, and prevents the execution of non-value-adding activities.



Our multi-agent system follows a fully software-agnostic approach, integrating seamlessly into existing engineering toolchains. Deployment options are flexible, offering both cloud-based and fully on-premises solutions to meet client needs. In sectors such as aerospace and defense, on-premises implementations are essential to comply with strict data sovereignty requirements. By automating routine tasks across engineering domains, engineers are empowered to focus on product evaluation and actual value creation. The multi-agent system accelerates time to market, improves product quality, reduces effort, and can be tailored to customer-specific data and requirements.

Lucas Kempe
Senior Manager
Accenture

Aleksandar Trenchev
Engineering Manager
Accenture

Till Haunschild
Associate Manager
Accenture

Jayas Jacob
Eng. Senior Analyst
Accenture

Vipin Neekamparambath
Eng. Senior Analyst
Accenture

The Future of Engineering

Engineering is at a turning point. AI is not only transforming individual tools or methods but also reshaping the organizational elements that determine the success of complex product development. We use the Accenture Butterfly Model (see Figure 28) as a template to discuss the effects of AI on four key organizational elements:

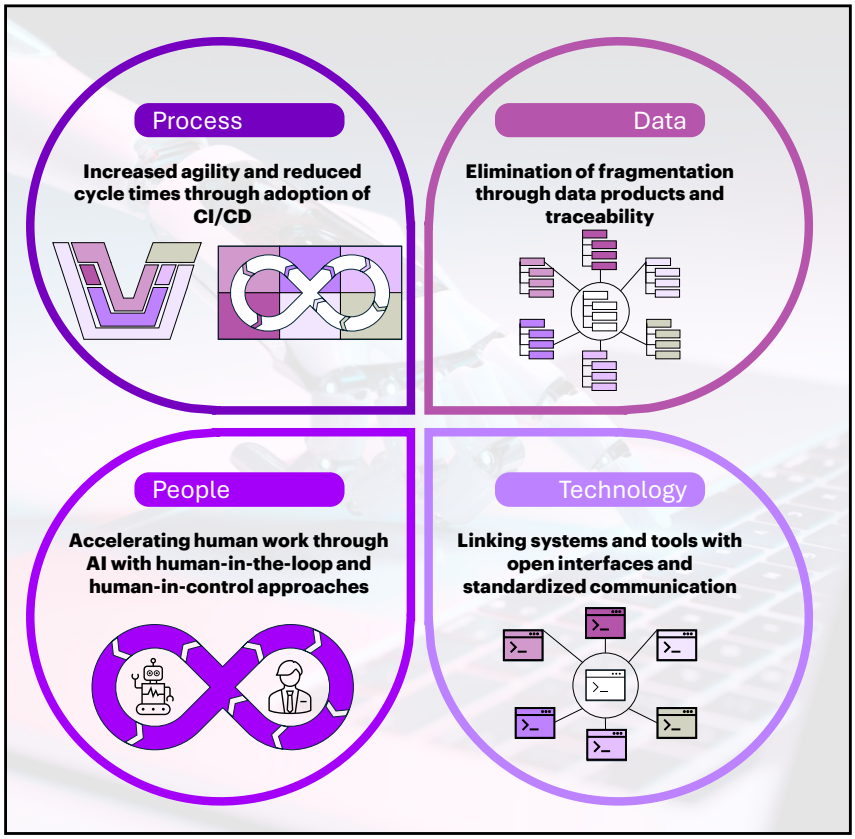


Figure 28: The Accenture Butterfly Model as a basis for assessing the impact of AI on organizational elements in engineering

Process

Processes are being reimagined and transformed to accelerate development cycles and improve quality. AI will massively accelerate product development processes. While individual engineering domains will benefit from AI-driven automation and optimization, the larger impact will stem from horizontal AI use cases that are applied across engineering domains. The enrichment of agentic systems with system-wide context will drive a fundamental shift from document-centric to model-based product development (MBSE), enabling horizontal integration across engineering disciplines. This transformation will lead to three key effects:

1. Tighter integration of domains and disciplines (mechanical, E/E, software), enabling continuous compatibility checks and thereby the early correction of inconsistencies (Zhang et al. 2025e).
2. Massive reduction of iteration cycle times, with a shift from the traditional V-model towards a CI/CD-inspired approach, significantly increasing agility. Expansion of design space exploration, as accelerated iterations allow multiple design alternatives to be developed in parallel (Zampetti et al. 2023).

3. As a result, critical design decisions can be deferred to later stages with higher product development maturity levels, fostering innovation and the exploration of previously untapped product variants (Zhang & Zhang 2025).

Beyond engineering itself, AI will enable a higher degree of connectivity across the entire enterprise process landscape. Feedback loops from manufacturing, supply chain, and service can be directly incorporated into development, creating a Closed Loop Engineering (Demartini et al. 2019) paradigm.

Data

Data becomes a strategic asset, providing the context, consistency and accessibility required for intelligent workflows. Data is the fuel for the digitalization and AI enablement of complex product development processes. In the future, companies will only succeed in developing advanced mechatronic systems if they master a set of data-driven capabilities:

1. Alignment of hierarchical data architectures with the engineering toolchain, ensuring that information flows seamlessly across tools and disciplines.
2. Reduction of the number of data formats and tools while simultaneously driving the standardization of data and interfaces.
3. Continuous consolidation, documentation, and cataloging of consumable and machine-readable data products, making them accessible for AI use cases (Jahnke & Otto 2023).
4. Ensuring data compliance with regulatory requirements and defined data models, for example, through automated policy enforcement.
5. Realization of MBSE and the connection of standardized data models with meta- and system models (Zhang et al. 2025e).
6. Application of AI not only for horizontal and vertical use cases, but also for the preparation, cleansing, and enrichment of datasets themselves (Singh 2023).

In this paradigm, data evolves from being a byproduct of engineering activities to a strategic enabler of AI-driven development. Organizations that successfully industrialize their data management practices will gain a decisive competitive advantage in the next generation of AI-enabled product engineering.

People

People are at the center, as adaptability, talent development, and close collaboration between humans and machines are key to success (Shao et al. 2025). AI will not replace human work in product development but rather complement and significantly accelerate it (Brynjolfsson et al. 2025).

Historically, the evolution of product creation has been shaped by rising complexity. In the shift from craftsmanship to mass production and later to increasingly complex mechatronic systems, organizations attempted to master this complexity by decomposing product development into smaller subprocesses. As a result, engineers have transformed from generalists in early manufacturing into highly specialized experts with narrow but demanding areas of responsibility.

AI enablement marks a turning point in this trajectory. In the future, the ability to design products via prompt-driven development will democratize product creation. Creative tasks will gain importance, while administrative and operational work will diminish. Faster realization of product ideas and deeper exploration of design spaces will allow engineers to develop unconventional product concepts to higher maturity levels and compare them with

conventional designs (Jiang et al. 2024). This will lead to increased innovation capacity and a shift in the boundaries of what is technically feasible.

At the same time, Human-in-the-Loop and Human-in-Control will remain essential. While many administrative and operational activities offer potential for automation, engineers must be empowered to evaluate and approve plausibility, validity, reliability, safety, and compliance of AI-generated content. This requires training and change management on the responsible and reliable use of AI, ensuring explainability, and embedding human checkpoints into digitized workflows (Lee et al. 2025). Ultimately, accountability for faulty designs cannot be delegated to algorithms or agents, it will remain with organizations and individuals.

Technology

Technology delivers platforms, software and architectures that enable AI-driven innovation and embed it sustainably within the organization. Over the coming years, the very platforms and tools that underpin engineering will themselves be profoundly reshaped by AI enablement.

Through the democratization of product development, user interfaces, as illustrated in Figure 26, will be consolidated into unified platforms. Tools will increasingly be operated via prompts, meaning that individual tools may no longer require stand-alone user interfaces but instead provide their functionality through integrated services (Riche et al. 2025). As a result, the focus for tool vendors will shift toward delivering open and high-performance input/output interfaces (e.g., APIs, MCP servers) that allow access to modular and configurable data architectures. These criteria will become decisive factors in the evaluation and selection of tools and vendors.

AI enablement will also transform data representations and storage. Automated generation of knowledge graphs and vector databases based on development data stored within platforms and tools will provide the contextual foundation required by agentic systems interacting directly with these tools. At the same time, built-in data quality monitoring systems will continuously oversee data structures, enforce policies, and ensure reliability, machine-readability, and completeness.

Together, these developments will redefine the role of technology in engineering. The tool landscape will evolve from a set of isolated tools into an intelligent, interconnected ecosystem that empowers AI-driven product development.

From Vision to Execution

The previous Sections have shown which dimensions should be considered in an engineering AI transformation and that there is currently an imbalance between vertical and horizontal AI use cases. Based on these findings, hypotheses have been formulated as to the direction in which elements of engineering (processes, data, people, technology) will develop. Finally, this Subsection presents a high-level roadmap with successive steps that will transform this process from a vision into execution. The roadmap is divided into six steps that describe the transformation process from a fragmented tool and data landscape to an AI-native engineering toolchain, as shown in Figure 29.

Figure 29: Proposed roadmap for achieving an AI-native engineering toolchain

1. **Define the North Star & Establish Alignment:** The first step is to reach a consensus on the long-term goal of AI transformation. Stakeholders develop a shared understanding

of the transformative power and role of AI and, based on this, draw up a strategic plan with milestones and identify quick wins. The quick wins are collected, structured, and clustered in the form of a use case funnel and include various AI use cases across the entire product development process. At the same time, the current status of the engineering toolchain, data landscape (data sources, data flows, data architectures, data catalogs), processes, and organizational structure is analyzed, and weaknesses that can be remedied with moderate effort are identified.

2. **Build the Foundation & Define Architecture:** In the second phase, the foundations for an AI transformation are established. This involves assigning strategic tools to the product development process and sorting out legacy tools that do not meet the requirements of AI-driven engineering in terms of interoperability (APIs) and standardization (certificates & data formats). The aim is to prepare tools, data, processes, and people for AI-driven engineering, reduce complexity, and eliminate redundancies. Decisions also need to be made regarding the governance framework and the selection of the AI platform. The use case funnel defined in the first phase is refined and the use cases to be implemented are specified. It is particularly important to ensure that a balance is struck between vertical and horizontal AI use cases and that the use cases can be integrated with each other in the future to enable consistency throughout the entire product development process.

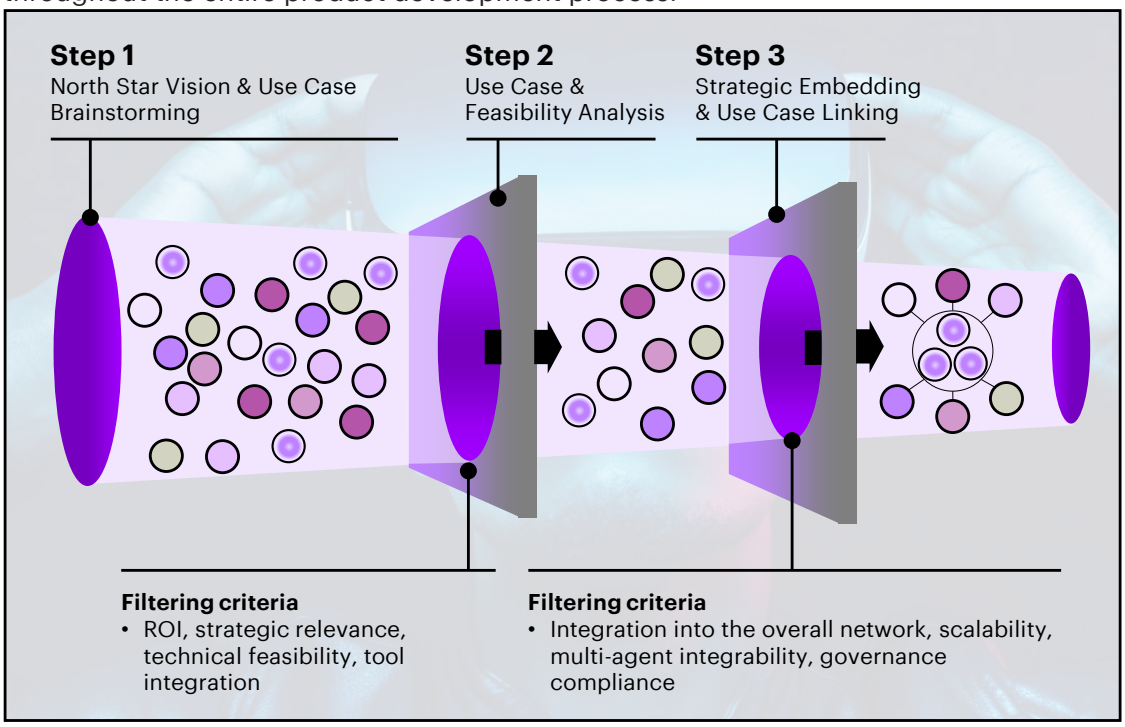


Figure 30: Strategic use case selection approach considering horizontal and vertical balance

Figure 30 shows an approach for selecting AI use cases in engineering, taking various criteria into account. In the first step, different use cases are identified, the number of which is then incrementally reduced until, in the end, a homogeneous network consisting of horizontal and vertical AI use cases results. These use cases are both technically feasible and offer a quick ROI but can also be connected to each other within multi-agent systems.

3. **Implement High-Value AI Use Cases:** In the third phase, the AI use cases are implemented and the five dimensions of the framework presented in Section *Framework for Scalable AI in Engineering* are continuously monitored. With the implementation of the use cases, data quality is improved, metrics for data quality are established and monitored, and consistent metadata management is introduced for each use case. Tool

interoperability is ensured through the implementation of MCP servers, and context modules in the form of vector and graph databases are introduced for each use case as needed. Compliance with the defined governance framework is also ensured by introducing a rights and roles concept for the AI platform that guarantees needs-based access to data and AI use cases.

4. **Scale, Connect and Contextualize:** As the implementation of AI use cases progresses, greater emphasis is being placed on interconnection and contextualizing AI use cases. In the fourth phase, the context modules will therefore be linked together, connections between use case-specific graph and vector databases will be established, and the goal of system-wide interconnection of data, processes, and tools will be pursued. System-wide interoperability of use cases is ensured through connections to MBSE tools, while domain-specific teams are given the opportunity to retrain the models in order to increase the performance of the use cases. The AI platform focuses on system-wide, automated monitoring in line with the defined governance framework and the establishment of feedback loops for the continuous improvement of individual use cases and their interoperability.
5. **Automate, Reason & Orchestrate:** In the fifth phase, the focus is on further interconnection of the use cases by setting up the multi-agent system. The goal in this phase is to create the higher-level agent layers from Figure 27 and to intelligently control the domain-specific use cases at the lower level (executed by executors from Figure 27). This enables the system-wide integration of the agentic AI capabilities from Figure 26 into the product development process and allows the introduction of cross-domain reasoning and orchestration capabilities. Cross-domain engineering processes such as change and configuration management can thus be supported and accelerated by AI, with the validity of the respective results of individual agents being ensured by human control mechanisms and decision as well as approval gates (human in control). The implementation of higher-level agent layers significantly increases the degree of automation in product development and enables state-dependent, adaptive workflows.
6. **Continuous Improvement & North Star Realization:** In the final phase, continuous improvements are implemented in order to converge towards the defined North Star vision. The number of connected tools is expanded, new use cases are implemented, and further automation is pursued with regard to interoperability, governance, and self-learning systems. Interfaces to production and feedback loops from production, use, maintenance, and logistics are also analyzed in this phase so that engineering is embedded in the company's entire AI ecosystem.

Comment on AI Adoption in Engineering

One cornerstone of the transformation toward AI-native engineering lies in turning today's fragmented data and tool landscape into an integrated, interoperable digital foundation. This transformation requires structuring engineering data for machine readability, adopting open APIs, and applying standards such as MCP and A2A to enable seamless cross-domain collaboration between AI agents. Unified namespaces and common data models (e.g., ISO 10303 STEP, OPC UA, SysML v2) are critical to ensure consistent interpretation and exchange of information across systems, domains and disciplines.

Together with a clear governance framework and the adoption of new ways of working within an AI operating model, this digital foundation provides the backbone for scaling AI in engineering. It enables organizations to move beyond isolated proof-of-concepts predominantly vertical integrated toward a scalable, domain-spanning application of AI that drives automation, generates measurable value, and fosters innovation across the entire engineering lifecycle.

Kathrin Schwan
Lead AI & Data DACH
Accenture


Summary

Companies engaged in complex product development are increasingly striving to automate their R&D processes and transform them through the integration of AI. This transformation promises substantial acceleration of product development cycles, higher product quality, improved compatibility between mechanical, E/E and software components, and enhanced capabilities for design space exploration. While numerous approaches for embedding AI into engineering already exist, most organizations have not yet established a comprehensive management framework or achieved large-scale deployment of this technology.

This white paper introduces a framework for scalable AI applications in engineering, designed to address the unique challenges of product development environments. Considering the specific boundary conditions in engineering, such as fragmented tools and data landscapes, heterogeneous data formats, stringent governance requirements, complex tool interoperability, and high interdependencies between engineering domains along the V-model, the framework identifies key dimensions that must be addressed to ensure scalable and sustainable AI integration. Following terminology from Systems Engineering, the framework distinguishes between vertical and horizontal AI use cases, depending on their level of domain specificity and cross-domain applicability.

Based on an extensive literature review covering AI use cases across all domains of the V-model, the paper highlights that most existing AI applications currently exhibit a high vertical maturity but limited horizontal integration. In other words, they are typically designed around specific tools and data sources within isolated domains, with insufficient focus on cross-domain networking and knowledge sharing. This pattern mirrors the current state of industrial AI adoption, which is often constrained by tool and data fragmentation as well as organizational silos. Consequently, the paper argues that AI transformation must be driven by top management, ensuring a balanced portfolio of vertical and horizontal use cases and promoting integration across domains to unlock system-wide benefits.

A further key insight of the study is the emerging role of multi-agent systems in engineering, which enable higher levels of automation and coordination between AI-driven tasks. This white paper illustrates how such systems can be applied in future engineering environments and analyzes their impact across four dimensions, namely processes, data, people, and technology. Finally, a roadmap is presented that outlines the path toward scalable AI deployment in product development.

In conclusion, the paper recommends a strategic and iterative approach to AI transformation: selecting and developing use cases in alignment with the proposed framework, progressively interconnecting them into multi-agent systems, and ensuring governance and scalability from the outset. To achieve lasting success, organizations must also make deliberate choices regarding the right tools and technologies and understand the correct sequence for generating and structuring the required engineering artifacts. Providing contextual information across different system levels is essential to enable consistent interpretation and automated reasoning. Furthermore, the long-term integrability of initially developed use cases must be safeguarded to ensure they can evolve into interconnected multi-agent ecosystems rather than remain isolated solutions. Finally, human checkpoints embedded into agent-driven workflows play a pivotal role in maintaining oversight, trust, and accountability, ensuring that automation augments rather than replaces engineering expertise.

References

Abboush, M. et al. (2024). Representative real-time dataset generation based on automated fault injection and hil simulation for ml-assisted validation of automotive software systems. *Electronics*, *13*(2), 437. **https://doi.org/10.3390/electronics13020437**

Abboush, M. et al. (2025). Advancing real-time validation of automotive software systems via continuous integration and intelligent failure analysis. *Scientific Reports*, *15*(1), 32936. **https://doi.org/10.1038/s41598-025-21416-5**

Abdalla, A. et al. (2024). Generative artificial intelligence for model-based graphical programming in automotive function development. *Available at SSRN 5153452*. **https://dx.doi.org/10.2139/ssrn.5153452**

Abdel-Aty, T. A., & Negri, E. (2024). Conceptualizing the digital thread for smart manufacturing: A systematic literature review. Journal of Intelligent Manufacturing, 35(8), 3629-3653. **https://doi.org/10.1007/s10845-024-02407-1**

Accenture Research Report (2021). *Thread-First Thinking: Staying in front of the immense wave of product data.* **https://www.accenture.com/us-en/insights/industry-x/thread-first-thinking** [Accessed: 25.10.25]

Accenture Research Report (2024). Going for growth – Navigating the great value migration in the age of AI. **https://www.accenture.com/us-en/insights/strategy/ai-enabled-growth** [Accessed: 25.10.25]

Admass, W. S. et al. (2024). Cyber security: State of the art, challenges and future directions. Cyber Security and Applications, 2, 100031. **https://doi.org/10.1016/j.csa.2023.100031**

Ahmad, K. et al. (2023). Requirements engineering framework for human-centered artificial intelligence software systems. Applied Soft Computing, 143, 110455. **https://doi.org/10.1016/j.asoc.2023.110455**

AI Marketplace (2022). Are your Engineering IT Standards ready for AI? (White Paper). **https://www.prostep.org/shop/detail?ai%5Baction%5D=detail&ai%5Bcontroller%5D=Catalog&ai%5Bd_name%5D=wp_kim_2022&ai%5Bd_pos%5D=** [Accessed: 25.10.2025]

Akhtar, S. et al. (2025). LLM-based event log analysis techniques: A survey. arXiv preprint arXiv:2502.00677. **https://doi.org/10.48550/arXiv.2502.00677**

Akundi, A. et al. (2024). Text-to-Model Transformation: Natural Language-Based Model Generation Framework. *Systems* 12, no. 9: 369. **https://doi.org/10.3390/systems12090369**

Alagarsamy, S. et al. (2024). Enhancing large language models for text-to-testcase generation. arXiv preprint arXiv:2402.11910. **https://doi.org/10.1016/j.jss.2025.112531**

Alsaif, K. M. et al. (2024). Multimodal Large Language Model-Based Fault Detection and Diagnosis in Context of Industry 4.0. Electronics, 13(24), 4912. **https://doi.org/10.3390/electronics13244912**

Altendeitering, M., & Guggenberger, T. M. (2024). Data Quality Tools: Towards a Software Reference Architecture. In HICSS (pp. 6159-6168). **10.24251/HICSS.2024.740**

Altendeitering, M. et al. (2024). A design theory for data quality tools in data ecosystems: Findings from three industry cases. Data & Knowledge Engineering, 153, 102333. **https://doi.org/10.1016/j.datak.2024.102333**

Alturayeif, N. et al. (2025). Machine learning approaches for automated software traceability: A systematic literature review. *Journal of Systems and Software*, 112536. **https://doi.org/10.1016/j.jss.2025.112536**

Amyan, A. et al. (2024). Automating Fault Test Cases Generation and Execution for Automotive Safety Validation via NLP and HIL Simulation. Sensors, 24(10), 3145. **https://doi.org/10.3390/s24103145**

Armbrust, M. et al. (2021). Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of CIDR (Vol. 8, p. 28). **https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf** [Accessed: 25.10.25]

Arora, C. et al. (2024a). Advancing requirements engineering through generative ai: Assessing the role of llms. In Generative AI for Effective Software Development (pp. 129-148). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-55642-5_6**

Arora, C. et al. (2024b). Towards standards-compliant assistive technology product specifications via llms. In 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW) (pp. 385-389). IEEE. **https://doi.org/10.1109/REW61692.2024.00060**

Auer, A. et al. (2025). TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning. arXiv preprint arXiv:2505.23719. **https://doi.org/10.48550/arXiv.2505.23719**

Azzabi, S. et al. (2024). Data lakes: A survey of concepts and architectures. Computers, 13(7), 183. **https://doi.org/10.3390/computers13070183**

Badagabettu, A. et al. (2024). Query2cad: Generating cad models using natural language queries. arXiv preprint arXiv:2406.00144. **https://doi.org/10.48550/arXiv.2406.00144**

Bader, E. et al. (2024). Facilitating User-Centric Model-Based Systems Engineering Using Generative AI. In MODELSWARD (pp. 371-377). **https://doi.org/10.5220/0012623200003645**

Bai, J. et al. (2024). A dynamic knowledge graph approach to distributed self-driving laboratories. Nature Communications, 15(1), 462. **https://doi.org/10.1038/s41467-023-44599-9**

Barnett, S. et al. (2024). Seven failure points when engineering a retrieval augmented generation system. In Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI (pp. 194-199). **https://doi.org/10.1145/3644815.3644945**

Bashir, S. et al. (2025). Requirements Ambiguity Detection and Explanation with LLMs: An Industrial Study. **https://www.ipr.mdu.se/pdf_publications/7221.pdf** [Accessed: 25.10.25]

Berriche, A. et al. (2020). Towards model synchronization for consistency management of mechatronic systems. Applied Sciences, 10(10), 3577. **https://doi.org/10.3390/app10103577**

Bernijazov, R. et al. (2025). AI-Augmented Model-Based Systems Engineering. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 120(s1), 96-100. **https://doi.org/10.1515/zwf-2024-0123**

Bharadwaj, A. G., & Starly, B. (2022). Knowledge graph construction for product designs from large CAD model repositories. Advanced Engineering Informatics, 53, 101680. **https://doi.org/10.1016/j.aei.2022.101680**

Bianchini, D., et al. (2024). Digital thread for smart products: A survey on technologies, challenges and opportunities in service-oriented supply chains. IEEE Access. **https://doi.org/10.1109/ACCESS.2024.3454375**

Birchler, C. et al. (2023). Machine learning-based test selection for simulation-based testing of self-driving cars software. Empirical Software Engineering, 28(3), 71. **https://doi.org/10.1007/s10664-023-10286-y**

Bleisinger, O. & Eigner, M. (2025). KI-Anwendungen im Engineering: Neue Technologien, neue Chancen?. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 120(s1), 39-43. **https://doi.org/10.1515/zwf-2024-0173**

Blocklove, J. et al. (2023). Chip-chat: Challenges and opportunities in conversational hardware design. In 2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD) (pp. 1-6). IEEE. **https://doi.org/10.1109/MLCAD58807.2023.10299874**

Bode, J. et al. (2024). Toward avoiding the data mess: industry insights from data mesh implementations. IEEE Access, 12, 95402-95416. **https://doi.org/10.1109/ACCESS.2024.3417291**

Bone, M. et al. (2018). Toward an interoperability and integration framework to enable digital thread. Systems, 6(4), 46. **https://doi.org/10.3390/systems6040046**

Bonner, M. et al. (2024). LLM - based Approach to Automatically Establish Traceability between Requirements and MBSE. In INCOSE International Symposium (Vol. 34, No. 1, pp. 2542-2560). **https://doi.org/10.1002/iis2.13285**

Bordas, A. et al. (2024). What is generative in generative artificial intelligence? A design-based perspective. Research in Engineering Design, 35(4), 427-443. **https://doi.org/10.1007/s00163-024-00441-x**

Borovits, N. et al. (2025). On the Maturity of LLMOps Services Computing: An Industrial Study. In International Conference on Advanced Information Systems Engineering (pp. 310-317). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-94931-9_25**

Bouamra, Y. et al. (2025). SysTemp: A Multi-Agent System for Template-Based Generation of SysML v2. arXiv preprint arXiv:2506.21608.
https://doi.org/10.48550/arXiv.2506.21608

Bröcker, M. A. et al. (2024). Integration of large data based on HDF5 in a collaborative and multidisciplinary design environment, Part A: Methodology and Implementation. In AIAA SciTech 2024 Forum (p. 0482). **https://doi.org/10.2514/6.2024-0482**

Brynjolfsson, E. et al. (2025). Generative AI at work. The Quarterly Journal of Economics, 140(2), 889-942. **https://doi.org/10.1093/qje/qjae044**

Burggräf, P. et al. (2024). AI-artifacts in engineering change management–a systematic literature review. Research in Engineering Design, 35(2), 215-237.
https://doi.org/10.1007/s00163-023-00430-6

CADQuery (2024). A python parametric cad scripting framework.
https://cadquery.readthedocs.io/en/latest/ [Accessed: 26.10.25]

Cao, Y. et al. (2023). A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. *arXiv preprint arXiv:2303.04226*.
https://doi.org/10.48550/arXiv.2303.04226

Chandrasegaran, S. K. et al. (2013). The evolution, challenges, and future of knowledge representation in product design systems. Computer-aided design, 45(2), 204-228.
https://doi.org/10.1016/j.cad.2012.08.006

Chang, K. et al. (2023). Chipgpt: How far are we from natural language hardware design. arXiv preprint arXiv:2305.14019.
https://doi.org/10.48550/arXiv.2305.14019

Chang, C. C. et al. (2024). Lamagic: Language-model-based topology generation for analog integrated circuits. *arXiv preprint arXiv:2407.18269*.
https://doi.org/10.48550/arXiv.2407.18269

Chang, E. Y., & Geng, L. (2025). SagaLLM: Context Management, Validation, and Transaction Guarantees for Multi-Agent LLM Planning. arXiv preprint arXiv:2503.11951. **https://doi.org/10.48550/arXiv.2503.11951**

Chen, G. et al. (2023a). Typefly: Flying drones with large language model. *arXiv preprint arXiv:2312.14950*. **https://doi.org/10.48550/arXiv.2312.14950**

Chen, T. et al. (2023b). TRouter: thermal-driven PCB routing via nonlocal crisscross attention networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *42*(10), 3388-3401. **https://doi.org/10.1109/TCAD.2023.3243544**

Chen, Y. et al. (2024). MetaOpenFOAM: an LLM-based multi-agent framework for CFD. arXiv preprint arXiv:2407.21320. **https://doi.org/10.48550/arXiv.2407.21320**

Chen, Y. et al. (2025a). OptMetaOpenFOAM: Large Language Model Driven Chain of Thought for Sensitivity Analysis and Parameter Optimization based on CFD. arXiv preprint arXiv:2503.01273. **https://doi.org/10.48550/arXiv.2503.01273**

Chen, J. et al. (2025b). FaultGPT: Industrial Fault Diagnosis Question Answering System by Vision Language Models. arXiv preprint arXiv:2502.15481. **https://doi.org/10.48550/arXiv.2502.15481**

Cheng, H. et al. (2024). Generative AI for Requirements Engineering: A Systematic Literature Review. arXiv preprint arXiv:2409.06741. **https://doi.org/10.48550/arXiv.2409.06741**

Choi, S., & Jung, Y. (2025). Knowledge Graph Construction: Extraction, Learning, and Evaluation. Applied Sciences, 15(7), 3727. **https://doi.org/10.3390/app15073727**

Cibrián, E. et al. (2025). Ensuring Semantic Consistency in SysML v2 Models Through Metamodel-Driven Validation. IEEE Access. **https://doi.org/10.1109/ACCESS.2025.3587786**

Cong, Y. et al. (2025). Enhancing novel product iteration: An integrated framework for heuristic ideation via interpretable conceptual design knowledge graph. Advanced Engineering Informatics, 65, 103131. **https://doi.org/10.1016/j.aei.2025.103131**

Dai, F. et al. (2025). State of the Art in Parallel and Distributed Systems: Emerging Trends and Challenges. Electronics, 14(4), 677. **https://doi.org/10.3390/electronics14040677**

Daneshyan, F. et al. (2025). SmartNote: An LLM-Powered, Personalised Release Note Generator That Just Works. arXiv preprint arXiv:2505.17977. **https://doi.org/10.1145/3729345**

Darm, P. et al. (2025). Inference-Time Intervention in Large Language Models for Reliable Requirement Verification. arXiv preprint arXiv:2503.14130. **https://doi.org/10.48550/arXiv.2503.14130**

Dassault Systèmes (2024). AI-Driven Generative Experiences - Succeed with AI in Knowledge-Based Engineering. **https://www.3ds.com/products/catia/ai-driven-generative-experiences** [Accessed: 26.10.25]

Demartini, M., et al. (2019). Closed-loop manufacturing for aerospace industry: An integrated PLM-MOM solution to support the wing box assembly process. In Advances in Production Management Systems. Towards Smart Production Management Systems: IFIP WG 5.7 International Conference, APMS 2019, Austin, TX, USA, September 1–5, 2019, Proceedings, Part II (pp. 423-430). Springer International Publishing. **https://doi.org/10.1007/978-3-030-29996-5_49**

Dong, Z. et al. (2025). Fine-tuning a large language model for automating computational fluid dynamics simulations. Theoretical and Applied Mechanics Letters, 100594. **https://doi.org/10.48550/arXiv.2504.09602**

Durão, L. F. C. et al. (2024). Digital twin data architecture for product-service systems. Procedia CIRP, 121, 79-84. **https://doi.org/10.1016/j.procir.2023.09.232**

Eigner, M. (2021). System Lifecycle Management: Engineering Digitalization (Engineering 4.0). Springer Nature. **https://doi.org/10.1007/978-3-662-62183-7**

Eken, B. et al. (2024). A multivocal review of MLOps practices, challenges and open issues. ACM Computing Surveys. **https://doi.org/10.1145/3747346**

El Asad, A. et al. (2025). Advancing Automotive Production: An LLM-based Impact Analysis for Software Updates. Procedia CIRP, 134, 127-132. **https://doi.org/10.1016/j.procir.2025.02.134**

El-Hajjami, A., & Salinesi, C. (2025). Synthline: A Product Line Approach for Synthetic Requirements Engineering Data Generation Using Large Language Models. In International Conference on Research Challenges in Information Science (pp. 208-225). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-92474-3_13**

Elrefaie, M. et al. (2025). AI Agents in Engineering Design: A Multi-Agent Framework for Aesthetic and Aerodynamic Car Design. arXiv preprint arXiv:2503.23315. **https://doi.org/10.48550/arXiv.2503.23315**

Esposito, M. et al. (2025). Generative ai for software architecture. applications, challenges, and future directions. Journal of Systems and Software, 112607. **https://doi.org/10.1016/j.jss.2025.112607**

Etemadi, K. et al. (2025). LLM-based Property-based Test Generation for Guardrailing Cyber-Physical Systems. arXiv preprint arXiv:2505.23549. **https://doi.org/10.48550/arXiv.2505.23549**

Ettinger, A. (2025). Enterprise Architecture as a Dynamic Capability for Scalable and Sustainable Generative AI adoption: Bridging Innovation and Governance in Large Organisations. arXiv preprint arXiv:2505.06326. **https://doi.org/10.48550/arXiv.2505.06326**

Failla, L. et al. (2025). Managing lifecycle of product information with an ontology-based knowledge framework. Journal of Industrial Information Integration, 45, 100820. **https://doi.org/10.1016/j.jii.2025.100820**

Fantechi, A. et al. (2023). Inconsistency detection in natural language requirements using chatgpt: a preliminary evaluation. In 2023 IEEE 31st International Requirements Engineering Conference (RE) (pp. 335-340). IEEE. **https://doi.org/10.1109/RE57278.2023.00045**

Faubel, L., & Schmid, K. (2024). MLOps: A Multiple Case Study in Industry 4.0. In 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 01-08). IEEE. **https://doi.org/10.1109/ETFA61755.2024.10711136**

Feng, J. et al. (2025). OpenFOAMGPT 2.0: end-to-end, trustworthy automation for computational fluid dynamics. arXiv preprint arXiv:2504.19338. **https://doi.org/10.48550/arXiv.2504.19338**

Ferrari, A., & Spoletini, P. (2025). Formal requirements engineering and large language models: A two-way roadmap. Information and Software Technology, 181, 107697. **https://doi.org/10.1016/j.infsof.2025.107697**

Ferrero, V. et al. (2022). Classifying component function in product assemblies with graph neural networks. Journal of Mechanical Design, 144(2), 021406. **https://doi.org/10.1115/1.4052720**

Feuerriegel, S. et al. (2024). Generative AI. *Business & Information Systems Engineering*, *66*(1), 111-126. **https://doi.org/10.1007/s12599-023-00834-7**

Foidl, H. et al. (2024). Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers. Journal of Systems and Software, 207, 111855. **https://doi.org/10.1016/j.jss.2023.111855**

Fragkoulis, M. et al. (2024). A survey on the evolution of stream processing systems. The VLDB Journal, 33(2), 507-541. **https://doi.org/10.1007/s00778-023-00819-8**

Fuchß, D. et al. (2025a). LiSSA: Toward Generic Traceability Link Recovery through Retrieval-Augmented Generation. In Proceedings of the IEEE/ACM 47th International Conference on Software Engineering. ICSE (Vol. 25). **https://doi.org/10.1109/ICSE55347.2025.00186**

Fuchß, D. et al. (2025b). Beyond Retrieval: A Study of Using LLM Ensembles for Candidate Filtering in Requirements Traceability. In *2025 IEEE 33rd International Requirements Engineering Conference Workshops (RE)*. **https://doi.org/10.1109/REW66121.2025.00006**

Gao, D. et al. (2024). Diffcad: Weakly-supervised probabilistic cad model retrieval and alignment from an rgb image. ACM Transactions on Graphics (TOG), 43(4), 1-15. **https://doi.org/10.1145/3658236**

Gärtner, A. E., & Göhlich, D. (2024). Automated requirement contradiction detection through formal logic and LLMs. Automated Software Engineering, 31(2), 49. **https://doi.org/10.1007/s10515-024-00452-x**

Gerhard, D. et al. (2025). Optimierung von Entwicklungsprozessen durch KI-gestütztes Generatives Engineering und Design: Ein methodisches Vorgehensmodell. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 120(s1), 70-75. **https://doi.org/10.1515/zwf-2024-0140**

Geyer, W. et al. (2025). A Case Study Investigating the Role of Generative AI in Quality Evaluations of Epics in Agile Software Development. In *Proceedings of the 4th Annual Symposium on Human-Computer Interaction for Work* (pp. 1-18). **https://doi.org/10.48550/arXiv.2505.07664**

Ghosh, S. et al. (2025). Digital twin, digital thread, and digital mindset in enabling digital transformation: A socio-technical systems perspective. Technovation, 144, 103240. **https://doi.org/10.1016/j.technovation.2025.103240**

Goedegebuure, A. et al. (2024). Data mesh: a systematic gray literature review. ACM Computing Surveys, 57(1), 1-36. **https://doi.org/10.1145/3687301**

Gu, Z. et al. (2024). A systematic overview of data federation systems. Semantic Web, 15(1), 107-165. **https://doi.org/10.3233/SW-223201**

Guan, Y. et al. (2025). CAD-Coder: Text-to-CAD Generation with Chain-of-Thought and Geometric Reward. arXiv preprint arXiv:2505.19713. **https://doi.org/10.48550/arXiv.2505.19713**

Guo, D. et al. (2024). DeepSeek-Coder: When the Large Language Model Meets Programming--The Rise of Code Intelligence. arXiv preprint arXiv:2401.14196. **https://doi.org/10.48550/arXiv.2401.14196**

Habiba, U. E. et al. (2024). How mature is requirements engineering for AI-based systems? A systematic mapping study on practices, challenges, and future research directions. *Requirements Engineering*, 1-34. **https://doi.org/10.1007/s00766-024-00432-3**

Hajisharifi, A. et al. (2024). An LSTM-enhanced surrogate model to simulate the dynamics of particle-laden fluid systems. Computers & Fluids, 280, 106361. **https://doi.org/10.1016/j.compfluid.2024.106361**

Han, H. et al. (2024a). Archcode: Incorporating software requirements in code generation with large language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 13520-13552). **https://doi.org/10.48550/arXiv.2408.00994**

Han, H. et al. (2024b). Retrieval-augmented generation with graphs (GraphRAG). arXiv preprint arXiv:2501.00309. **https://doi.org/10.48550/arXiv.2501.00309**

Han, Y. et al. (2025). Standard Applicability Judgment and Cross-jurisdictional Reasoning: A RAG-based Framework for Medical Device Compliance. arXiv preprint arXiv:2506.18511. **https://doi.org/10.48550/arXiv.2506.18511**

Hanke, F. et al. (2025). AI-augmented systems engineering: conceptual application of retrieval-augmented generation for model-based systems engineering graph. Proceedings of the Design Society, 5, 439-448. **https://doi.org/10.1017/pds.2025.10058**

Harby, A. A., & Zulkernine, F. (2025). Data lakehouse: a survey and experimental study. Information Systems, 127, 102460. **https://doi.org/10.1016/j.is.2024.102460**

Hassani, S. (2024). Enhancing legal compliance and regulation analysis with large language models. In 2024 IEEE 32nd International Requirements Engineering Conference (RE) (pp. 507-511). IEEE. **https://doi.org/10.1109/RE59067.2024.00065**

Hassine, J. (2024). An LLM-based approach to recover traceability links between security requirements and goal models. In Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (pp. 643-651). **https://doi.org/10.1145/3661167.3661261**

Haug, S. et al. (2025). Automated Code Generation and Validation for Software Components of Microcontrollers. *arXiv preprint arXiv:2502.18905*. **https://doi.org/10.48550/arXiv.2502.18905**

Hedberg Jr, T. D., et al. (2020). Using graphs to link data across the product lifecycle for enabling smart manufacturing digital threads. Journal of Computing and Information Science in Engineering, 20(1), 011011. **https://doi.org/10.1115/1.4044921**

Heidari, N., & Iosifidis, A. (2024). Geometric deep learning for computer-aided design: A survey. arXiv preprint arXiv:2402.17695. **https://doi.org/10.48550/arXiv.2402.17695**

Hemmat, A. et al. (2025). Research directions for using LLM in software requirement engineering: a systematic review. Frontiers in Computer Science, 7, 1519437. **https://doi.org/10.3389/fcomp.2025.1519437**

Herrmann, L., & Kollmannsberger, S. (2024). Deep learning in computational mechanics: a review. Computational Mechanics, 74(2), 281-331. **https://doi.org/10.1007/s00466-023-02434-4**

Hey, T. et al. (2024). Requirements classification for traceability link recovery. In 2024 IEEE 32nd International Requirements Engineering Conference (RE) (pp. 155-167). IEEE. **https://doi.org/10.1109/RE59067.2024.00024**

Hey, T. et al. (2025). Requirements Traceability Link Recovery via Retrieval-Augmented Generation. In *International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 381-397). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-88531-0_27**

Hoang, D. et al. (2025). Knowledge Graph Fusion with Large Language Models for Accurate, Explainable Manufacturing Process Planning. arXiv preprint arXiv:2506.13026. **https://doi.org/10.48550/arXiv.2506.13026**

Holterman, E. et al. (2024). Roadmap to Strengthen the US Manufacturing Supply Chain via Digital Thread Technology. **https://www.nist.gov/publications/roadmap-strengthen-us-manufacturing-supply-chain-digital-thread-technology** [Accessed: 26.10.25]

Hooshmand, Y. et al. (2022). From a monolithic PLM landscape to a federated domain and data mesh. *Proceedings of the Design Society*, *2*, 713-722. **https://doi.org/10.1017/pds.2022.73**

Hou, S. et al. (2025a). AutoFEA: Enhancing AI Copilot by Integrating Finite Element Analysis Using Large Language Models with Graph Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 22, pp. 24078-24085). **https://doi.org/10.1609/aaai.v39i22.34582**

Hou, X. et al. (2025b). Model context protocol (mcp): Landscape, security threats, and future research directions. arXiv preprint arXiv:2503.23278. **https://doi.org/10.48550/arXiv.2503.23278**

Hovemann, A. et al. (2025). Prompt Engineering im Systems Engineering: Potenziale und Grenzen moderner großer Sprachmodelle im V-Modell. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 120(s1), 101-106. **https://doi.org/10.1515/zwf-2024-0139**

Huang, D. et al. (2023). Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. arXiv preprint arXiv:2312.13010. **https://doi.org/10.48550/arXiv.2312.13010**

Hur, A. et al. (2021). A survey on state-of-the-art techniques for knowledge graphs construction and challenges ahead. In 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE) (pp. 99-103). IEEE. **https://doi.org/10.1109/AIKE52691.2021.00021**

Hurni, T. et al. (2021). Complementor dedication in platform ecosystems: rule adequacy and the moderating role of flexible and benevolent practices. European Journal of Information Systems, 30(3), 237-260. **https://doi.org/10.1080/0960085X.2020.1779621**

Ibrahim, N. et al. (2024). A survey on augmenting knowledge graphs (KGs) with large language models (LLMs): models, evaluation metrics, benchmarks, and challenges. Discover Artificial Intelligence, 4(1), 76. **https://doi.org/10.1007/s44163-024-00175-8**

Ishida, S. et al. (2024). Langprop: A code optimization framework using large language models applied to driving. arXiv preprint arXiv:2401.10314. **https://doi.org/10.48550/arXiv.2401.10314**

Ismail, F. N. et al. (2025). Big Data Architecture for Large Organizations. arXiv preprint arXiv:2505.04717. **https://doi.org/10.48550/arXiv.2505.04717**

Jahnke, N., & Otto, B. (2023). Data catalogs in the enterprise: applications and integration. Datenbank-Spektrum, 23(2), 89-96. **https://doi.org/10.1007/s13222-023-00445-2**

Jamieson, L. et al. (2024). A review of deep learning methods for digitisation of complex documents and engineering diagrams. Artificial Intelligence Review, 57(6), 136. **https://doi.org/10.1007/s10462-024-10779-2**

Jarrahi, M. H. et al. (2023). The principles of data-centric AI. Communications of the ACM, 66(8), 84-92. **https://doi.org/10.1145/3571724**

Jiang, T. et al. (2024). Human-AI interaction research agenda: A user-centered perspective. Data and Information Management, 8(4), 100078. **https://doi.org/10.1016/j.dim.2024.100078**

Jiang, P. et al. (2025). A two-stage retrieval-augmented generation framework for producing sustainable product design guidelines. Sustainable Futures, 10, 101094.
https://doi.org/10.1016/j.sftr.2025.101094

Jin, H. et al. (2024). From llms to llm-based agents for software engineering: A survey of current, challenges and future. arXiv preprint arXiv:2408.02479.
https://doi.org/10.48550/arXiv.2408.02479

Jin, X. et al. (2025). A Closed-Loop Multi-Agent Framework for Aerodynamics-Aware Automotive Styling Design. arXiv preprint arXiv:2508.03370.
https://doi.org/10.48550/arXiv.2508.03370

Jnini, A. et al. (2025). Physics-constrained deeponet for surrogate cfd models: a curved backward-facing step case. arXiv preprint arXiv:2503.11196.
https://doi.org/10.48550/arXiv.2503.11196

Johns, B. et al. (2024). AI Systems Modeling Enhancer (AI - SME): Initial Investigations into a ChatGPT - enabled MBSE Modeling Assistant. In INCOSE International Symposium (Vol. 34, No. 1, pp. 1149-1168). **https://doi.org/10.1002/iis2.13201**

Jones, B. T. et al. (2023). Self-supervised representation learning for cad. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 21327-21336).
https://doi.org/10.1109/CVPR52729.2023.02043

Jung, M. et al. (2024). ContrastCAD: Contrastive Learning-Based Representation Learning for Computer-Aided Design Models. IEEE Access, 12, 84830-84842.
https://doi.org/10.1109/ACCESS.2024.3415816

Kang, N. (2025). Generative AI-driven design optimization: eight key application scenarios. JMST Advances, 1-7. **https://doi.org/10.1007/s42791-025-00097-1**

Kang, S. et al. (2025). Explainable automated debugging via large language model-driven scientific debugging. Empirical Software Engineering, 30(2), 1-28.
https://doi.org/10.1007/s10664-024-10594-x

Karagoz, E. et al. (2024). Identification of Missing Knowledge in MBSE System Models Using Graph - Based Machine Learning. Systems Engineering, e70013.
https://doi.org/10.1002/sys.70013

Kasper, N. et al. (2024). The digital thread for system lifecycle management with a native graph database in a polyglot architecture. Proceedings of the Design Society, 4, 2079-2088. **https://doi.org/10.1017/pds.2024.210**

Khan, M. T. H., & Rezwana, S. (2021). A review of CAD to CAE integration with a hierarchical data format (HDF)-based solution. Journal of King Saud University-Engineering Sciences, 33(4), 248-258. **https://doi.org/10.1016/j.jksues.2020.04.009**

Khoee, A. G. et al. (2024). GoNoGo: An Efficient LLM-based Multi-Agent System for Streamlining Automotive Software Release Decision-Making. In IFIP International Conference on Testing Software and Systems (pp. 30-45). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-80889-0_3**

Kirchner, S., & Knoll, A. C. (2025). Generating Automotive Code: Large Language Models for Software Development and Verification in Safety-Critical Systems. arXiv preprint arXiv:2506.04038. **https://doi.org/10.48550/arXiv.2506.04038**

Kolt, N. (2025). Governing AI agents. arXiv preprint arXiv:2501.07913. **https://doi.org/10.48550/arXiv.2501.07913**

Kommineni, V. K. et al. (2024). From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. arXiv preprint arXiv:2403.08345. **https://doi.org/10.48550/arXiv.2403.08345**

Koziolek, H. et al. (2024). Automated control logic test case generation using large language models. In 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1-8). IEEE. **https://doi.org/10.1109/ETFA61755.2024.10711016**

Kreuzberger, D. et al. (2023). Machine learning operations (MLOps): Overview, definition, and architecture. IEEE access, 11, 31866-31879. **https://doi.org/10.1109/ACCESS.2023.3262138**

Kumar, A. et al. (2024). LLMs as Evaluators: A Novel Approach to Evaluate Bug Report Summarization. arXiv preprint arXiv:2409.00630. **https://doi.org/10.48550/arXiv.2409.00630**

Kumar, M. et al. (2025). Your synthetic teammate: Enriching new product development with generative AI. Business Horizons. **https://doi.org/10.1016/j.bushor.2025.02.008**

Kunc, O., & Bröcker, M. A. (2024). Integration of large data based on HDF5 in a collaborative and multidisciplinary design environment, Part B: Application to Structural Mechanics of Gas Turbine Engines. In AIAA Scitech 2024 Forum (p. 0383). **https://doi.org/10.2514/6.2024-0383**

Kwon, S. et al. (2020). Enriching standards-based digital thread by fusing as-designed and as-inspected data using knowledge graphs. Advanced Engineering Informatics, 46, 101102. **https://doi.org/10.1016/j.aei.2020.101102**

Lameh, J. et al. (2025). Automating Feature Modeling in Product Line Engineering for Systems Engineering: The Application of Natural Language Processing. In 2nd Workshop on Model-based System Engineering and Artificial Intelligence (pp. 450-457). SCITEPRESS-Science and Technology Publications. **https://doi.org/10.5220/0013442900003896**

Larichev, V. et al. (2025). Generative AI and Agentic Architecture in Engineering and Manufacturing: Potentials and Practice of Scalable AI Solutions. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 120(s1), 17-24. **https://doi.org/10.1515/zwf-2024-0166**

Lebioda, K. et al. (2025). Are requirements really all you need? A case study of LLM-driven configuration code generation for automotive simulations. arXiv preprint arXiv:2505.13263. **https://doi.org/10.48550/arXiv.2505.13263**

LeCun, Y. et al. (2015). Deep learning. *nature*, *521*(7553), 436-444. **https://doi.org/10.1038/nature14539**

Lee, J., & Su, H. (2023). A unified industrial large knowledge model framework in industry 4.0 and smart manufacturing. arXiv preprint arXiv:2312.14428. **https://doi.org/10.48550/arXiv.2312.14428**

Lee, J. et al. (2025). Engineering Artificial Intelligence: Framework, Challenges, and Future Direction. *arXiv preprint arXiv:2504.02269*. **https://doi.org/10.48550/arXiv.2504.02269**

Li, J. et al. (2025a). Cad-llama: Leveraging large language models for computer-aided design parametric 3d model generation. In Proceedings of the Computer Vision and Pattern Recognition Conference (pp. 18563-18573). **https://doi.org/10.48550/arXiv.2505.04481**

Li, S. et al. (2025b). Enhancing Retrieval-Augmented Generation: A Study of Best Practices. arXiv preprint arXiv:2501.07391.
https://doi.org/10.48550/arXiv.2501.07391

Li, K. Y. et al. (2025c). Generative AI and CAD automation for diverse and novel mechanical component designs under data constraints. Discover Applied Sciences, 7(4), 1-21.
https://doi.org/10.1007/s42452-025-06833-5

Li, S. et al. (2025d). Collaborative Inference and Learning between Edge SLMs and Cloud LLMs: A Survey of Algorithms, Execution, and Open Challenges. arXiv preprint arXiv:2507.16731.
https://doi.org/10.48550/arXiv.2507.16731

Li, X. et al. (2025e). LLM4CAD: Multimodal Large Language Models for Three-Dimensional Computer-Aided Design Generation. Journal of Computing and Information Science in Engineering, 25(2), 021005. **https://doi.org/10.1115/1.4067085**

Li, M. et al. (2025f). Specllm: Exploring generation and review of vlsi design specification with large language model. In 2025 International Symposium of Electronics Design Automation (ISEDA) (pp. 749-755). IEEE. **https://doi.org/10.1109/ISEDA65950.2025.11100410**

Li, Z. et al. (2025g). From system 1 to system 2: A survey of reasoning large language models. arXiv preprint arXiv:2502.17419.
https://doi.org/10.48550/arXiv.2502.17419

Liang, K. et al. (2024a). A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12), 9456-9478. **https://doi.org/10.1109/TPAMI.2024.3417451**

Liang, X. et al. (2024b). A survey of LLM-augmented knowledge graph construction and application in complex product design. Procedia CIRP, 128, 870-875.
https://doi.org/10.1016/j.procir.2024.07.069

Liang, X. et al. (2025). A survey of large language model-augmented knowledge graphs for advanced complex product design. Journal of Manufacturing Systems, 80, 883-901.
https://doi.org/10.1016/j.jmsy.2025.04.016

Liepert, C. et al. (2024). Digital Twin Data Provision Within Engineering: An AAS PLM Implementation Ensuring Interoperability. In International Conference on Subject-Oriented Business Process Management (pp. 3-23). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-72041-3_1**

Liu, M. et al. (2023a). Chipnemo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*. **https://doi.org/10.48550/arXiv.2311.00176**

Liu, V. et al. (2023b). 3DALL-E: Integrating text-to-image AI in 3D design workflows. In *Proceedings of the 2023 ACM designing interactive systems conference* (pp. 1955-1977). **https://doi.org/10.1145/3563657.3596098**

Liu, M. et al. (2024a). An empirical study of the code generation of safety-critical software using llms. Applied Sciences, 14(3), 1046. **https://doi.org/10.3390/app14031046**

Liu, Y. et al. (2024b). Are LLMs good at structured outputs? A benchmark for evaluating structured output capabilities in LLMs. Information Processing & Management, 61(5), 103809. **https://doi.org/10.1016/j.ipm.2024.103809**

Liu, Y. et al. (2025). LLM-ACNC: Aerospace Requirement Texts Knowledge Graph Construction Utilizing Large Language Model. Aerospace, 12(6), 463. **https://doi.org/10.3390/aerospace12060463**

Loconte, D. et al. (2024). Expanding the cloud-to-edge continuum to the IoT in serverless federated learning. Future Generation Computer Systems, 155, 447-462. **https://doi.org/10.1016/j.future.2024.02.024**

Longshore, R. et al. (2024). Leveraging Generative AI to Modify and Query MBSE Models. Acquisition Research Program. **https://dair.nps.edu/handle/123456789/5237** [Accessed: 26.10.25]

Lubos, S. et al. (2024). Leveraging LLMs for the quality assurance of software requirements. In 2024 IEEE 32nd International Requirements Engineering Conference (RE) (pp. 389-397). IEEE. **https://doi.org/10.1109/RE59067.2024.00046**

Luo, H. et al. (2024). Large language model-based code generation for the control of construction assembly robots: A hierarchical generation approach. *Developments in the Built Environment*, *19*, 100488. **https://doi.org/10.1016/j.dibe.2024.100488**

Luo, H. et al. (2025). Graph-R1: Towards Agentic GraphRAG Framework via End-to-end Reinforcement Learning. arXiv preprint arXiv:2507.21892. **https://doi.org/10.48550/arXiv.2507.21892**

Luttmer, J. et al. (2021). Representation and application of digital standards using knowledge graphs. Proceedings of the Design Society, 1, 2551-2560. **https://doi.org/10.1017/pds.2021.516**

Ma, R. et al. (2024a). Verilogreader: Llm-aided hardware test generation. In *2024 IEEE LLM Aided Design Workshop (LAD)* (pp. 1-5). IEEE. **https://doi.org/10.48550/arXiv.2406.04373**

Ma, Z. et al. (2024b). Llmparser: An exploratory study on using large language models for log parsing. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (pp. 1-13). **https://doi.org/10.48550/arXiv.2404.18001**

Madireddy, S. et al. (2025). Large Language Model-Driven Code Compliance Checking in Building Information Modeling. Electronics, 14(11), 2146. **https://doi.org/10.48550/arXiv.2506.20551**

Mahadevkar, S. V. et al. (2024). Exploring AI-driven approaches for unstructured document analysis and future horizons. Journal of Big Data, 11(1), 92. **https://doi.org/10.1186/s40537-024-00948-z**

Maharjan, R. et al. (2023). Benchmarking message queues. In Telecom (Vol. 4, No. 2, pp. 298-312). MDPI. **https://doi.org/10.3390/telecom4020018**

Majigi, M. U. et al. (2025). Big data transfer service architecture for cloud data centers: problems, methods, applications, and future trends. *Discover Computing*, *28*(1), 1-41. **https://doi.org/10.1007/s10791-025-09682-3**

Masoudifard, A. et al. (2024). Leveraging Graph-RAG and Prompt Engineering to Enhance LLM-Based Automated Requirement Traceability and Compliance Checks. arXiv preprint arXiv:2412.08593. **https://doi.org/10.48550/arXiv.2412.08593**

Massoudi, S., & Fuge, M. (2025). Agentic Large Language Models for Conceptual Systems Engineering and Design. arXiv preprint arXiv:2507.08619. **https://doi.org/10.48550/arXiv.2507.08619**

Mavromatis, C., & Karypis, G. (2024). GNN-RAG: Graph neural retrieval for large language model reasoning. arXiv preprint arXiv:2405.20139. **https://doi.org/10.48550/arXiv.2405.20139**

Mehlstäubl, J. et al. (2022). Using machine learning for product portfolio management: a methodical approach to predict values of product attributes for multi-variant product portfolios. Proceedings of the Design Society, 2, 1659-1668. **https://doi.org/10.1017/pds.2022.168**

Mehmood, Y. et al. (2024). MLOps critical success factors-A systematic literature review. VFAST Transactions on Software Engineering, 12(1), 183-209. **https://doi.org/10.21015/vtse.v12i1.1747**

Mei, L. et al. (2025). A Survey of Context Engineering for Large Language Models. arXiv preprint arXiv:2507.13334. **https://doi.org/10.48550/arXiv.2507.13334**

Meng, Y., & Ban, A. (2024). Automated UML Class Diagram Generation from Textual Requirements Using NLP Techniques. JOIV: International Journal on Informatics Visualization, 8(3-2), 1905-1915. **http://dx.doi.org/10.62527/joiv.8.3-2.3482**

Milchevski, D. et al. (2025). Multi-Step Generation of Test Specifications using Large Language Models for System-Level Requirements. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track) (pp. 132-146). **https://doi.org/10.18653/v1/2025.acl-industry.11**

Mishra, S. et al. (2024). An AI-driven data mesh architecture enhancing decision-making in infrastructure construction and public procurement. arXiv preprint arXiv:2412.00224. **https://doi.org/10.48550/arXiv.2412.00224**

Möltner, T. et al. (2025). Creation, Evaluation and Self-Validation of Simulation Models with Large Language Models. (PrePrint) Research Square. **https://doi.org/10.21203/rs.3.rs-6566994/v1**

Mohammed, S. et al. (2025). The effects of data quality on machine learning performance on tabular data. *Information Systems*, *132*, 102549. **https://doi.org/10.1016/j.is.2025.102549**

Mottaghian, S. et al. (2025). Scheitert Systems Engineering an seiner eigenen Komplexität? Wie KI die operative Produktentwicklung beflügelt. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 120(s1), 90-95. **https://doi.org/10.1515/zwf-2025-0008**

Muttillo, V. et al. (2024). Towards synthetic trace generation of modeling operations using in-context learning approach. In Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (pp. 619-630). **https://doi.org/10.1145/3691620.3695058**

Naeem, Z. A. et al. (2024). RetClean: Retrieval-Based Data Cleaning Using LLMs and Data Lakes. Proceedings of the VLDB Endowment, 17(12), 4421-4424. **https://doi.org/10.14778/3685800.3685890**

Narayan, A. et al. (2022). Can foundation models wrangle your data?. arXiv preprint arXiv:2205.09911. **https://doi.org/10.48550/arXiv.2205.09911**

Nau, S. et al. (2025). SPICEAssistant: LLM using SPICE Simulation Tools for Schematic Design of Switched-Mode Power Supplies. *arXiv preprint arXiv:2507.10639*. **https://doi.org/10.48550/arXiv.2507.10639**

Ni, B. et al. (2025a). Towards trustworthy knowledge graph reasoning: An uncertainty aware perspective. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 12, pp. 12417-12425). **https://doi.org/10.48550/arXiv.2410.08985**

Ni, J. et al. (2025b). CADDesigner: Conceptual Design of CAD Models Based on General-Purpose Agent. *arXiv preprint arXiv:2508.01031*. **https://doi.org/10.48550/arXiv.2508.01031**

Nielsen, M. K. et al. (2024). Industrial R&D project portfolio selection method using a multi-objective optimization program: A conceptual quantitative framework. *Journal of Industrial Engineering and Management*, *17*(1), 217-234. **https://doi.org/10.3926/jiem.6552**

Ning, F. et al. (2025). A review and assessment of 3D CAD model retrieval in machine-part design. International Journal of Computer Integrated Manufacturing, 38(6), 752-774. **https://doi.org/10.1080/0951192X.2024.2382196**

Niu, F. et al. (2025). TVR: Automotive System Requirement Traceability Validation and Recovery Through Retrieval-Augmented Generation. arXiv preprint arXiv:2504.15427. **https://doi.org/10.48550/arXiv.2504.15427**

Norheim, J. et al. (2024). Challenges in applying large language models to requirements engineering tasks. Design Science, 10, e16. **https://doi.org/10.1017/dsj.2024.8**

Nouri, A. et al. (2024). Engineering safety requirements for autonomous driving with large language models. In 2024 IEEE 32nd International Requirements Engineering Conference (RE) (pp. 218-228). IEEE. **https://doi.org/10.1109/RE59067.2024.00029**

Nouri, A. et al. (2025). On Simulation-Guided LLM-based Code Generation for Safe Autonomous Driving Software. *arXiv preprint arXiv:2504.02141*. **https://doi.org/10.48550/arXiv.2504.02141**

Ocker, F. et al. (2025). From idea to CAD: a language model-driven multi-agent system for collaborative design. arXiv preprint arXiv:2503.04417. **https://doi.org/10.48550/arXiv.2503.04417**

Otto, B. (2011). Data governance. *Business & Information Systems Engineering*, *3*(4), 241-244. **https://doi.org/10.1007/s12599-011-0162-8**

Pahune, S. et al. (2025). The Importance of AI Data Governance in Large Language Models. Big Data and Cognitive Computing, 9(6), 147. **https://doi.org/10.3390/bdcc9060147**

Paliwal, G. et al. (2024). Accelerating time-to-market: the role of generative AI in product development. In 2024 IEEE Colombian Conference on Communications and Computing (COLCOM) (pp. 1-9). IEEE. **https://doi.org/10.1109/COLCOM62950.2024.10720255**

Pan, J. et al. (2025). A survey of research in large language models for electronic design automation. ACM Transactions on Design Automation of Electronic Systems, 30(3), 1-21. **https://doi.org/10.1145/3715324**

Pandey, S. et al. (2025). OpenFOAMGPT: A retrieval-augmented large language model (LLM) agent for OpenFOAM-based computational fluid dynamics. Physics of Fluids, 37(3). **https://doi.org/10.1063/5.0257555**

Panta, N. P. et al. (2025). MEDA: A Multi-Agent System For Parametric CAD Model Creation. **https://www.researchgate.net/publication/394977611_MEDA_A_Multi-Agent_System_For_Parametric_CAD_Model_Creation** [Accessed: 26.10.25]

Patel, A. et al. (2024). Easing adoption of model based system engineering with application of generative ai. In 2024 IEEE Space, Aerospace and Defence Conference (SPACE) (pp. 871-874). IEEE. **https://doi.org/10.1109/SPACE63117.2024.10667868**

Patil, M. S. et al. (2024). Towards specification-driven LLM-based generation of embedded automotive software. In International Conference on Bridging the Gap between AI and Reality (pp. 125-144). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-75434-0_9**

Peng, B. et al. (2024). Graph retrieval-augmented generation: A survey. arXiv preprint arXiv:2408.08921. **https://doi.org/10.48550/arXiv.2408.08921**

Petrovic, N. et al. (2025). Survey of GenAI for Automotive Software Development: From Requirements to Executable Code. arXiv preprint arXiv:2507.15025. **https://doi.org/10.48550/arXiv.2507.15025**

Picard, C. et al. (2025). From concept to manufacturing: Evaluating vision-language models for engineering design. Artificial Intelligence Review, 58(9), 288. **https://doi.org/10.1007/s10462-025-11290-y**

Plettenberg, P. et al. (2025). Graph Neural Networks for Automatic Addition of Optimizing Components in Printed Circuit Board Schematics. *arXiv preprint arXiv:2506.10577*. **https://doi.org/10.48550/arXiv.2506.10577**

Poulsen, V. V. et al. (2025). Advancing systems engineering with artificial intelligence: a review on the future potential, challenges and pathways. Proceedings of the Design Society, 5, 359-368. **https://doi.org/10.1017/pds.2025.10050**

prostep ivip (2025). Smart Systems Engineering – Collaborative Simulation-Based Engineering. Prostep ivip SmartSE Recommendation 2025 PSI 11. **https://www.prostep.org/fileadmin/prod-pay-download-8c1d/PSI11_RecV4_Final_aw.pdf** [Accessed: 26.10.25]

PTC (2024). What Manufacturers Need to Know About Generative Design - A technology whitepaper for executives. **https://www.ptc.com/en/resources/cad/ebook/what-manufacturers-need-to-know-about-generative-design** [Accessed: 26.10.25]

Pu, Y. et al. (2024). Customized retrieval augmented generation and benchmarking for EDA tool documentation QA. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design* (pp. 1-9). **https://doi.org/10.1145/3676536.3676730**

Qin, S. et al. (2024). Intelligent design and optimization system for shear wall structures based on large language models and generative artificial intelligence. Journal of Building Engineering, 95, 109996. **https://doi.org/10.1016/j.jobe.2024.109996**

Qin, F. et al. (2025). CADGCL: unsupervised retrieval of CAD models via boundary representations. The Visual Computer, 1-13. **https://doi.org/10.1007/s00371-025-03949-y**

Quan, Y. et al. (2024). Self-supervised Graph Neural Network for Mechanical CAD Retrieval. arXiv preprint arXiv:2406.08863. **https://doi.org/10.48550/arXiv.2406.08863**

Ray, P. P. (2025). A Review on Agent-to-Agent Protocol: Concept, State-of-the-art, Challenges and Future Directions. Authorea Preprints. **https://doi.org/10.36227/techrxiv.174612014.42157096/v1**

Reinpold, L. M. et al. (2024). Exploring LLMs for Verifying Technical System Specifications Against Requirements. In 2024 IEEE 3rd Industrial Electronics Society Annual On-Line Conference (ONCON) (pp. 1-6). IEEE. **https://doi.org/10.1109/ONCON62778.2024.10931625**

Riche, N. et al. (2025). AI-Instruments: Embodying Prompts as Instruments to Abstract & Reflect Graphical Interface Commands as General-Purpose Tools. In Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (pp. 1-18). **https://doi.org/10.1145/3706598.3714259**

Ronanki, K. et al. (2023). Investigating ChatGPT's potential to assist in requirements elicitation processes. In 2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 354-361). IEEE.
https://doi.org/10.1109/SEAA60479.2023.00061

Ryś, A. et al. (2024). Model management to support systems engineering workflows using ontology-based knowledge graphs. Journal of Industrial Information Integration, 42, 100720.
https://doi.org/10.1016/j.jii.2024.100720

Rzig, D. E. et al. (2024). Empirical Analysis on CI/CD Pipeline Evolution in Machine Learning Projects. arXiv preprint arXiv:2403.12199.
https://doi.org/10.48550/arXiv.2403.12199

Sadri-Moshkenani, Z. et al. (2022). Survey on test case generation, selection and prioritization for cyber - physical systems. Software Testing, Verification and Reliability, 32(1), e1794. **https://doi.org/10.1002/stvr.1794**

SAE (2014). Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems (On-Road Automated Vehicle Standards Committee). SAE Standard J, 3016, 1.
https://www.sae.org/standards/j3016_202104-taxonomy-definitions-terms-related-driving-automation-systems-road-motor-vehicles [Accessed: 26.10.25]

Said, A. et al. (2023). Circuit design completion using graph neural networks. *Neural Computing and Applications*, *35*(16), 12145-12157. **https://doi.org/10.1007/s00521-023-08346-x**

Saleem, S. et al. (2025). PassionNet: An Innovative Framework for Duplicate and Conflicting Requirements Identification. Expert Systems with Applications, 128684.
https://doi.org/10.1016/j.eswa.2025.128684

Samhan, A. et al. (2024). A Review of AI-Assisted Impact Analysis for Software Requirements Change: Challenges and Future Directions. In 2024 25th International Arab Conference on Information Technology (ACIT) (pp. 1-13). IEEE.
https://doi.org/10.1109/ACIT62805.2024.10877072

Schmid, L. et al. (2025). Software Architecture Meets LLMs: A Systematic Literature Review. *arXiv preprint arXiv:2505.16697*.
https://doi.org/10.48550/arXiv.2505.16697

Schneider, J. et al. (2024). The lakehouse: State of the art on concepts and technologies. SN Computer Science, 5(5), 449. **https://doi.org/10.1007/s42979-024-02737-0**

Sevenhuijsen, M. et al. (2025). VeCoGen: Automating Generation of Formally Verified C Code with Large Language Models. In *2025 IEEE/ACM 13th International Conference on Formal Methods in Software Engineering (FormaliSE)* (pp. 101-112). IEEE. **https://doi.org/10.1109/FormaliSE66629.2025.00017**

Shao, Y. et al. (2025). Future of Work with AI Agents: Auditing Automation and Augmentation Potential across the US Workforce. arXiv preprint arXiv:2506.06576. **https://doi.org/10.48550/arXiv.2506.06576**

Shi, F. et al. (2025). A stepwise intelligence generative method for structured maintenance guidance documents based on knowledge graph augmented LLM. Advanced Engineering Informatics, 67, 103523. **https://doi.org/10.1016/j.aei.2025.103523**

Shin, S. et al. (2023). Topology optimization via machine learning and deep learning: a review. Journal of Computational Design and Engineering, 10(4), 1736-1766. **https://doi.org/10.1093/jcde/qwad072**

Singh, P. (2023). Systematic review of data-centric approaches in artificial intelligence and machine learning. Data Science and Management, 6(3), 144-157. **https://doi.org/10.1016/j.dsm.2023.06.001**

Siemens (2024). Generative Design. **https://www.sw.siemens.com/en-US/technology/generative-design/** [Accessed: 26.10.25]

Somasekharan, N. et al. (2025). CFD-LLMBench: A Benchmark Suite for Evaluating Large Language Models in Computational Fluid Dynamics. arXiv preprint arXiv:2509.20374. **https://doi.org/10.48550/arXiv.2509.20374**

Song, Q. et al. (2025). Synthetic versus real: an analysis of critical scenarios for autonomous vehicle testing. Automated Software Engineering, 32(2), 37. **https://doi.org/10.1007/s10515-025-00499-4**

Sovrano, F. et al. (2025). Simplifying software compliance: AI technologies in drafting technical documentation for the AI Act. Empirical Software Engineering, 30(3), 91. **https://doi.org/10.1007/s10664-025-10645-x**

Strittmatter, J. (2025). Retrieval-Augmented Generation (RAG): Strategies and possible Applications in Software Engineering. **https://doi.org/10.5445/IR/1000181162**

Subramanian, S. et al. (2025). Small language models (SLMs) can still pack a punch: A survey. arXiv preprint arXiv:2501.05465. **https://doi.org/10.48550/arXiv.2501.05465**

Sui, Y. et al. (2025). Stop overthinking: A survey on efficient reasoning for large language models. arXiv preprint arXiv:2503.16419. **https://doi.org/10.48550/arXiv.2503.16419**

Sultan, B., & Apvrille, L. (2024). AI-driven consistency of SysML diagrams. In Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (pp. 149-159). **https://doi.org/10.1145/3640310.3674079**

Sunil, P. & Sills, R. B. (2024). FE-PINNs: finite-element-based physics-informed neural networks for surrogate modeling. arXiv preprint arXiv:2412.07126. **https://doi.org/10.48550/arXiv.2412.07126**

Steffen, D. et al. (2025). Code the Product–Vision für die Produktentstehung der Zukunft. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 120(s1), 55-60. **https://doi.org/10.1515/zwf-2024-0172**

Tao, L. et al. (2024). LLM-R: A Framework for Domain-Adaptive Maintenance Scheme Generation Combining Hierarchical Agents and RAG. arXiv preprint arXiv:2411.04476. **https://doi.org/10.48550/arXiv.2411.04476**

Thakur, S. et al. (2023). Autochip: Automating hdl generation using llm feedback. *arXiv preprint arXiv:2311.04887*. **https://doi.org/10.48550/arXiv.2311.04887**

Theodorakopoulos, L. et al. (2024). A state-of-the-art review in big data management engineering: Real-life case studies, challenges, and future research directions. Eng, 5(3), 1266-1297. **https://doi.org/10.3390/eng5030068**

Tian, R. et al. (2024). Debugbench: Evaluating debugging capability of large language models. arXiv preprint arXiv:2401.04621. **https://doi.org/10.48550/arXiv.2401.04621**

Tikayat Ray, A. et al. (2024). Development of a language model for named-entity-recognition in aerospace requirements. Journal of Aerospace Information Systems, 21(6), 489-499. **https://doi.org/10.2514/1.I011251**

Timperley, L. R. et al. (2025). Assessment of large language models for use in generative design of model based spacecraft system architectures. Journal of Engineering Design, 1-21. **https://doi.org/10.1080/09544828.2025.2453401**

Tinnes, C. et al. (2024). From Unstructured Product Descriptions to Structured Data for Industry 4.0 with ChatGPT. In 2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS) (pp. 1-8). IEEE. **https://doi.org/10.1109/ICPS59941.2024.10639992**

Tong, G. et al. (2024). An improved model combining knowledge graph and GCN for PLM knowledge recommendation. Soft Computing, 28(6), 5557-5575. **https://doi.org/10.1007/s00500-023-09340-0**

Tran, K. T. et al. (2025). Multi-agent collaboration mechanisms: A survey of llms. arXiv preprint arXiv:2501.06322. **https://doi.org/10.48550/arXiv.2501.06322**

Treshcheva, E. et al. (2025). Test2Text: AI-Based Mapping between Autogenerated Tests and Atomic Requirements. In 2025 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 17-20). IEEE. **https://doi.org/10.1109/ICSTW64639.2025.10962519**

Tsai, Y. et al. (2024). Rtlfixer: Automatically fixing rtl syntax errors with large language model. In Proceedings of the 61st ACM/IEEE Design Automation Conference (pp. 1-6). **https://doi.org/10.1145/3649329.3657353**

Vaicenavičius, J. et al. (2025). SysIDE: SysML v2 textual editing and analysis system: overview and applications. CEAS Space Journal, 1-7. **https://doi.org/10.1007/s12567-025-00595-x**

VDI (2021). VDI/VDE 2206 „Entwicklung mechatronischer und cyber-physischer Systeme“. **https://www.vdi.de/richtlinien/programme-zu-vdi-richtlinien/vdi-2206** [Accessed: 26.10.25]

Vogelsang, A., & Fischbach, J. (2025). Using large language models for natural language processing tasks in requirements engineering: A systematic guideline. In Handbook on Natural Language Processing for Requirements Engineering (pp. 435-456). Cham: Springer Nature Switzerland. **https://doi.org/10.1007/978-3-031-73143-3_16**

Von Heissen, O. et al. (2024). Toward Intelligent Generation of System Architectures. DS 130: Proceedings of NordDesign 2024, Reykjavik, Iceland, 12th-14th August 2024, 504-513. **https://doi.org/10.35199/NORDDESIGN2024.54**

Von Scherenberg, F. et al. (2024). Data sovereignty in information systems. Electronic Markets, 34(1), 15. **https://doi.org/10.1007/s12525-024-00693-4**

Voria, G. et al. (2025). RECOVER: Toward Requirements Generation from Stakeholders' Conversations. IEEE Transactions on Software Engineering. **https://doi.org/10.1109/TSE.2025.3572056**

Wan, Y. et al. (2024). Making knowledge graphs work for smart manufacturing: Research topics, applications and prospects. Journal of manufacturing systems, 76, 103-132. **https://doi.org/10.1016/j.jmsy.2024.07.009**

Wang, J. et al. (2024a). Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering. **https://doi.org/10.1109/TSE.2024.3368208**

Wang, J. et al. (2024b). Large language models-guided dynamic adaptation for temporal knowledge graph reasoning. Advances in Neural Information Processing Systems, 37, 8384-8410. **https://doi.org/10.48550/arXiv.2405.14170**

Wang, L. et al. (2024c). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, *18*(6), 186345. **https://doi.org/10.1007/s11704-024-40231-1**

Wang, Y. et al. (2025a). From code generation to software testing: AI Copilot with context-based RAG. IEEE Software. **https://doi.org/10.1109/MS.2025.3549628**

Wang, Z. et al. (2025b). An LLM-enabled Multi-Agent Autonomous Mechatronics Design Framework. arXiv preprint arXiv:2504.14681. **https://doi.org/10.48550/arXiv.2504.14681**

Wawrzik, F. et al. (2025). KGG4SE: A Knowledge Graph Generation Framework for Systems Engineering. **https://www.semantic-web-journal.net/system/files/swj3844.pdf** [Accessed: 26.10.25]

Williams, C. K., & Karahanna, E. (2013). Causal explanation in the coordinating process: A critical realist case study of federated IT governance structures. Mis Quarterly, 933-964. **https://www.jstor.org/stable/43826007** [Accessed: 26.10.25]

Wu, H. et al. (2024a). Chateda: A large language model powered autonomous agent for eda. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *43*(10), 3184-3197. **https://doi.org/10.1109/TCAD.2024.3383347**

Wu, J. et al. (2024b). A comprehensive analysis of challenges and strategies for software release notes on GitHub. Empirical Software Engineering, 29(5), 104. **https://doi.org/10.1007/s10664-024-10486-0**

Wu, S. et al. (2025). Cognitive Digital Thread Tool-Chain for Model Versioning in Model-Based Systems Engineering. Advanced Engineering Informatics, 67, 103490. **https://doi.org/10.1016/j.aei.2025.103490**

Wynn-Williams, S. et al. (2025). Can Generative AI Produce Test Cases? An Experience from the Automotive Domain. In Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (pp. 456-467). **https://doi.org/10.1145/3696630.3728568**

Xiao, Y. et al. (2024). Stronger, cheaper and demonstration-free log parsing with LLMs. arXiv preprint arXiv:2406.06156. **https://doi.org/10.48550/arXiv.2406.06156**

Xiao, Y. et al. (2025). GraphRAG-Bench: Challenging Domain-Specific Reasoning for Evaluating Graph Retrieval-Augmented Generation. arXiv preprint arXiv:2506.02404. **https://doi.org/10.48550/arXiv.2506.02404**

Xie, X. et al. (2024). Knowledge graph-based in-context learning for advanced fault diagnosis in sensor networks. Sensors, 24(24), 8086. **https://doi.org/10.3390/s24248086**

Xiong, X. et al. (2025). DR-RAG: Domain-Rule-based Retrieval-Augmented Generation for aviation digital model design. Advanced Engineering Informatics, 68, 103688. **https://doi.org/10.1016/j.aei.2025.103688**

Xu, J. et al. (2024). Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. arXiv preprint arXiv:2411.04954. **https://doi.org/10.48550/arXiv.2411.04954**

Xu, T. et al. (2025a). AI-Integrated Framework for Enhancing High Level Architecture Design Across System Lifecycle Stages. In 13th International Conference on Model-Based Software and Systems Engineering. **https://www.scitepress.org/Papers/2025/134442/134442.pdf** [Accessed: 26.10.25]

Xu, W. et al. (2025b). LLM-Based Agents for Tool Learning: A Survey. Data Science and Engineering, 1-31. **https://doi.org/10.1007/s41019-025-00296-9**

Xu, X. et al. (2025c). RAGOps: Operating and Managing Retrieval-Augmented Generation Pipelines. *arXiv preprint arXiv:2506.03401*. **https://doi.org/10.48550/arXiv.2506.03401**

Yahya, M. et al. (2024). A benchmark dataset with Knowledge Graph generation for Industry 4.0 production lines. Semantic Web, 15(2), 461-479. **https://doi.org/10.3233/SW-233431**

Yang, W. et al. (2025). Impact and influence of modern AI in metadata management. arXiv preprint arXiv:2501.16605. **https://doi.org/10.48550/arXiv.2501.16605**

Yao, J. et al. (2022). Edge-cloud polarization and collaboration: A comprehensive survey for ai. IEEE Transactions on Knowledge and Data Engineering, 35(7), 6866-6886. **https://doi.org/10.1109/TKDE.2022.3178211**

Yao, X. et al. (2024). Hdldebugger: Streamlining hdl debugging with large language models. ACM Transactions on Design Automation of Electronic Systems. **https://doi.org/10.1145/3735638**

Yue, L. et al. (2025a). Foam-agent: Towards automated intelligent cfd workflows. arXiv preprint arXiv:2505.04997. **https://doi.org/10.48550/arXiv.2505.04997**

Yue, L. et al. (2025b). Foam-Agent 2.0: An End-to-End Composable Multi-Agent Framework for Automating CFD Simulation in OpenFOAM. arXiv preprint arXiv:2509.18178. **https://doi.org/10.48550/arXiv.2509.18178**

Zampetti, F. et al. (2023). Continuous integration and delivery practices for cyber-physical systems: An interview-based study. ACM Transactions on Software Engineering and Methodology, 32(3), 1-44. **https://doi.org/10.1145/3571854**

Zhan, S. et al. (2025). A Review on Federated Learning Architectures for Privacy-Preserving AI: Lightweight and Secure Cloud–Edge–End Collaboration. Electronics, 14(13), 2512. **https://doi.org/10.3390/electronics14132512**

Zhang, H. et al. (2023). Large language models as data preprocessors. arXiv preprint arXiv:2308.16361. **https://doi.org/10.48550/arXiv.2308.16361**

Zhang, Q. et al. (2024). A Literature Review of the Digital Thread: Definition, Key Technologies, and Applications. Systems, 12(3), 70. **https://doi.org/10.3390/systems12030070**

Zhang, H., & Zhang, R. (2025). Generative artificial intelligence (AI) in built environment design and planning–A state-of-the-art review. Progress in Engineering Science, 2(1), 100040. **https://doi.org/10.1016/j.pes.2024.100040**

Zhang, L. et al. (2025a). GenAI for Simulation Model in Model-Based Systems Engineering. arXiv preprint arXiv:2503.06422. **https://doi.org/10.48550/arXiv.2503.06422**

Zhang, L. et al. (2025b). Large language models for computer-aided design: A survey. arXiv preprint arXiv:2505.08137. **https://doi.org/10.48550/arXiv.2505.08137**

Zhang, Q. et al. (2025c). A survey of graph retrieval-augmented generation for customized large language models. arXiv preprint arXiv:2501.13958. **https://doi.org/10.48550/arXiv.2501.13958**

Zhang, Z. et al. (2025d). A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, *43*(6), 1-47. **https://doi.org/10.1145/3748302**

Zhang, L. et al. (2025e). MBSE 2.0: Toward More Integrated, Comprehensive, and Intelligent MBSE. *Systems*, *13*(7), 584. **https://doi.org/10.3390/systems13070584**

Zhang, W. et al. (2025f). MBSE Co-Pilot: A Research Roadmap. *Systems Engineering*, e70011. **https://doi.org/10.1002/sys.70011**

Zhang, X. et al. (2025g). Using large language models for parametric shape optimization. Physics of Fluids, 37(8). **https://doi.org/10.1063/5.0273363**

Zhang, Z. (2025h). Application of deep reinforcement learning in parameter optimization and refinement of turbulence models. Scientific Reports, 15(1), 25236. **https://doi.org/10.1038/s41598-025-00351-5**

Zhang, Q. et al. (2025i). Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv preprint arXiv:2510.04618*. **https://doi.org/10.48550/arXiv.2510.04618**

Zhao, P. et al. (2024). Retrieval-augmented generation for ai-generated content: A survey. arXiv preprint arXiv:2402.19473. **https://doi.org/10.48550/arXiv.2402.19473**

Zhong, A. et al. (2024). Logparser-llm: Advancing efficient log parsing with large language models. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 4559-4570). **https://doi.org/10.1145/3637528.3671810**

Zhou, B. et al. (2024a). CausalKGPT: Industrial structure causal knowledge-enhanced large language model for cause analysis of quality problems in aerospace product manufacturing. Advanced Engineering Informatics, 59, 102333. **https://doi.org/10.1016/j.aei.2023.102333**

Zhou, Y. et al. (2024b). A survey on data quality dimensions and tools for machine learning. arXiv preprint arXiv:2406.19614. **https://doi.org/10.48550/arXiv.2406.19614**

Zhou, Z. et al. (2025). CAD-Judge: Toward Efficient Morphological Grading and Verification for Text-to-CAD Generation. arXiv preprint arXiv:2508.04002. **https://doi.org/10.48550/arXiv.2508.04002**

Zhu, J. et al. (2024). Relational Data Cleaning Meets Artificial Intelligence: A Survey. Data Science and Engineering, 1-28. **https://doi.org/10.1007/s41019-024-00266-7**

Zhu, X. et al. (2025). Knowledge graph-guided retrieval augmented generation. arXiv preprint arXiv:2502.06864. **https://doi.org/10.48550/arXiv.2502.06864**

Zou, X. (2020). A survey on application of knowledge graph. In Journal of Physics: Conference Series (Vol. 1487, No. 1, p. 012016). IOP Publishing. **https://doi.org/10.1088/1742-6596/1487/1/012016**

List of Abbreviations

Abbreviation	Meaning
A2A	Agent-to-Agent Protocol
AI	Artificial Intelligence
BOM	Bill of Material
CAD	Computer-Aided Design
CASE	Computer-Aided Software Engineering
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DL	Deep Learning
E-CAD	Electrical Computer-Aided Design
EDA	Electronic Design Automation
FEA	Fixed Entity Architecture
FEM	Finite Element Method
FL	Federated Learning
GAN	Generative Adversarial Network
GenAI	Generative Artificial Intelligence
GNN	Graph Neural Network
GTO	Generative Topology Optimization
HiL	Hardware-in-the-Loop
HDL	Hardware Description Language
KG	Knowledge Graph
LLM	Large Language Model
LSTM	Long Short-Term Memory
MBSE	Model-based Systems Engineering
MCP	Model Context Protocol
M-CAD	Mechanical Computer-Aided Design
PCB	Printed Circuit Board
PDM	Product Data Management
PINN	Physics-Informed Neural Network
PLM	Product Lifecycle Management
PMT	Processes, Methods & Tools
RAG	Retrieval-Augmented Generation
RC	Resistor-Capacity
RTE	Register-Transfer Level
ROI	Return on Investment
SiL	Software-in-the-Loop
SLM	Small Language Model
SysML	System Modeling Language
UML	Unified Modeling Language

Authors

Dr.-Ing. Dirk Alexander Molitor

Engineering and AI Consultant at Accenture, focusing on data-driven product development and engineering toolchain transformation. His work focuses heavily on AI enablement trough data models, tool architectures, and strategic AI use case selection.

Vlad Larichev

Manager and Industrial AI Lead at Accenture, specializing in the deployment of AI-driven solutions across the product lifecycle. His work enhances industrial operations by architecting intelligent systems that harness real-time data and optimize processes.

Dr.-Ing. Tobias Guggenberger

Group Lead at Fraunhofer ISST, focusing on the management and governance of inter-organizational data and AI systems. His work advances trustworthy industrial data ecosystems for effective collaboration and value creation.

Dr.-Ing. Marcel Altendeitering

Head of the Mobility & Smart Cities department at Fraunhofer ISST, focusing on automated data quality management and the role of data quality for data ecosystems. His works supports organizations leveraging the full potential of data-intensive applications.

Dr.-Ing. Daniel Porta

Group Lead at DFKI, focusing on digital transformation and future applied industrial AI. His work centers on asset administration shells, interactive digital twins, agentic AI, and cognitive assistants supporting humans in Industry 4.0.

Dr. rer. nat. Matthias Ziegler

Managing Director for Emerging Technology Innovation at Accenture, driving the adoption of cutting-edge technologies such as generative AI, robotics, and cloud native platforms. His work accelerates business value by turning strategic R&D into scalable solutions that empower clients to lead in digital-transformation-driven markets.

Acknowledgements

The authors would like to express their sincere gratitude to all contributors and reviewers who supported the development of this white paper. We would like to thank Tobias Geißinger, Timmo Sturm, Daniel Spieß, Rüdiger Stern, Andreas Kiep, Sebastian Angerer, Max Haberstroh, Thomas Reisenweber, Christian Kohlschein, Christof Horn, Liam Friel, Garrett Graham, Ronobijay Bhaumik, Ashish Wadjikar, Georg Brutzer, Nitin Ugale, Alexia Solvay, Marcus Hammes, Atilla Akdere, Rick Bouter, Prashant Chouhan and Hendrik Purwins for the many stimulating discussions.

We would also like to thank the sponsors and program leadership of the Accenture x DFKI AI Partner Lab: Jochen Malinowski, Felix Klemm, Antonio Krüger, Galia Diez, Jens Haupert, Harun Karimpur, and Pascal Lessel.

We would also like to express our sincere thanks to Ann-Christine Predian, Alexander Kemper, Timo Altmann, Nina Böckel, Florian Böhme, Christopher Knorr, Serdar Bulut and Pascalis Trentsios for their valuable support. Their expertise and commitment were fundamental in shaping the quality and depth of this work.

Special thanks go to Benedict Homuth and Yannik Dahmann for their valuable support in preparing the white paper and contributing to the editorial.