

Large Language Models in Industry 5.0

– Multi-Case Experiences from the AIMS5.0 Project

Csaba Hegedűs¹, Péter Olaszi¹, Máté Bancsics¹, Dániel Ficzer^{1,2}, Alex Latipov³, Mike Holenderski³, Guntis V. Strazds⁴, Belal Abu Naim⁵, Yasin Ghafourian⁵, Markus Tauber⁵, Marta Bonik⁶, Chris Damkat⁷, David Dandolo⁸, Chiara Masiero⁸, Manuel Barusco⁹, Gian Antonio Susto⁹, Ruba Almahasneh¹, and Pál Varga¹

Abstract—The transition from Industry 4.0 toward the humancentered vision of Industry 5.0 creates new opportunities for AI systems that augment rather than replace human expertise. This paper presents the integration of Large Language Models (LLMs) and AI copilots into industrial engineering workflows through multiple cases within the EU AIMS5.0 project. Eight use cases span system-of-systems engineering, support and aftersales services, regulatory self-assessment, product engineering, factual correctness checking, robot programming, ontology and knowledge-graph applications, and AI-assisted testing. The cases range from conceptual designs and proofs of concept to pilot implementations and therefore provide implementation experience and design insights rather than directly comparable performance results. Across the cases, recurring architectural patterns include Retrieval-Augmented Generation (RAG), semantic search, multiagent planning, natural-language interfaces, code generation, and verification mechanisms. The experiences also reveal common deployment challenges related to heterogeneous and evolving data sources, multimodal content, access control, factual reliability, legacy-system integration, and evaluation. Viewed through the Industry 5.0 dimensions of human-centricity, sustainability, and resilience, the cases illustrate how LLM-based systems can support knowledge access, engineering assistance, system adaptation, and safer interaction with industrial AI. The findings emphasize that successful industrial adoption depends not only on model capabilities, but also on high-quality domain content, appropriate safeguards, transparent system behavior, integration with existing infrastructure, and continuous human oversight.

Index Terms—Large Language Models, Industry 5.0, Human-Centered AI, AI Copilots, Retrieval-Augmented Generation, Industrial AI, Intelligent Manufacturing, Human–AI Collaboration

¹ Department of Telecommunications and Artificial Intelligence, Budapest University of Technology and Economics, Műgyetem rkp. 3., H-1111 Budapest, Hungary.

² AI & IIoT Division, AITIA International Inc., 7 Záhony Street, 1031 Budapest, Hungary.

³ Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands.

⁴ Institute of Electronics and Computer Science, Dzērbenes iela 14, LV-1006, Rīga, Latvia.

⁵ Research Studios Austria Forschungsgesellschaft mbH, Smart Digital Industries and Services, Heinrich Schmidladl-Straße 15, A-3100 St. Pölten, Austria.

⁶ Infineon Technologies AG, Am Campeon 1–15, 85579 Neubiberg, Germany.

⁷ Signify Netherlands B.V., High Tech Campus, 5656AE Eindhoven, The Netherlands.

⁸ Statwolf Data Science S.r.l., Padova, Italy.

⁹ Department of Information Engineering, University of Padova, Italy.

I. INTRODUCTION AND MOTIVATION FOR LLMs IN INDUSTRIAL PRODUCTION

Industry 5.0 extends the digitalization and automation of Industry 4.0 with a stronger focus on human-centricity, sustainability, and resilience. In this setting, Artificial Intelligence (AI) is not only a means of automating industrial processes. It can also support engineers, operators, and service personnel in tasks that require access to complex information and interaction with technical systems. Large Language Models (LLMs), chatbots, and AI copilots are especially relevant because they provide a natural-language interface to knowledge, software, and engineering tools.

This role is important because industrial information is often fragmented. Engineers may need to consult technical manuals, specifications, maintenance records, project documentation, and internal databases before they can resolve a problem. The relevant material may also be distributed across systems, stored in different formats, or maintained in different versions. Finding and combining this information can therefore be slow and error-prone. LLM-based assistants can help by retrieving and summarizing relevant material, particularly when they are combined with Retrieval-Augmented Generation (RAG) and domain-specific knowledge sources. Their usefulness, however, depends on more than the language model itself. Data quality, access rights, factual reliability, and integration with existing infrastructure are equally important in industrial settings.

Recently many research works have explored LLMs for individual engineering and manufacturing tasks, including knowledge retrieval, code generation, robotic control, and production-system interaction. These studies show the breadth of possible applications, but industrial deployments differ greatly in maturity, autonomy, and available evidence. A broader view is therefore useful for understanding which implementation patterns recur across use cases and which challenges appear when LLMs are integrated into real engineering environments.

This paper reports experience from the EU AIMS5.0 project through a portfolio of industrial LLM applications. The cases span several stages of the industrial lifecycle and include engineering assistance, support services, compliance guidance, software and robot programming, knowledge management,

and AI evaluation. Their maturity ranges from conceptual architectures and proofs of concept to pilot implementations. For this reason, the aim of the paper is not to compare the cases statistically. Instead, we examine how different forms of LLM integration can support industrial workflows and how they relate to the goals of Industry 5.0.

The paper makes three main contributions. It first brings together implementation experience from several industrial LLM applications within one project. It then analyzes the cases through a common framework that considers the role of the LLM, the human–system relationship, maturity, available evidence, and relevance to Industry 5.0. Finally, it identifies recurring implementation patterns and practical challenges across the cases.

The remainder of the paper is organized as follows. Section II reviews related work on LLM technologies and industrial applications. Section III introduces the qualitative multi-case methodology and the analytical framework. Section IV presents the individual use cases and their implementation experience. Section V concludes the paper with the main observations, limitations, and directions for future work.

II. RELATED WORKS

A. Large language models and Agents

The introduction of the transformer architecture [1] started the epoch of Large Language Models (LLMs). LLMs can be categorized by their architecture as encoder-, decoder, or encoder-decoder-based. From an application standpoint, they can be completion, chat, or reasoning models. We can differentiate between the supported modalities as well.

Most state-of-the-art (SotA) LLMs utilize In-Context Learning (ICL) with prompt engineering, where models learn to perform new tasks by using examples provided in the input prompt without parameter updates [2]. Studies suggest [3], [4] that the emergent reasoning abilities of LLMs are enabled by ICL, though the exact mechanisms remain to be explained.

LLMs and their reasoning abilities can be utilized with creative prompt engineering. A single-task prompt is an instruction or query that requires a single, specific task for the Artificial Intelligence (AI) to complete. They are commonly used in various Natural Language Processing (NLP) applications, such as text classification, summarization, translation or Named Entity Recognition. These tasks are fundamental to many AI applications, including chatbots, search engines, automated content generation, and data analysis [5], [6].

Prompt engineering assumes that effective prompts are built on specific core elements. The effect of these elements is measurable even for smaller, fine-tuned models: adding instructional text to the input raised the accuracy of a Hungarian BERT model on a language-understanding benchmark from 65% to 85% [7]. These include instructions which outline the task the model needs to perform; context, which offers external information to guide the model's understanding; input data, representing the questions or problems requiring solutions; and output indicators, which define the expected format or type of response [8], [9].

More complicated tasks require a chain of thought and/or a chain of interactions with external services. In general LLMs

struggle with complex reasoning. To address these challenges the LLM can be guided to reason over multi-step solutions. Chain-of-Thought (CoT) [10] prompting, for example, helps LLMs break down complex tasks into smaller steps. The limitations of CoT can be addressed by various decomposition strategies, increasing context memory, and adaptive step-by-step processing, such as Tree of Thoughts [11]. CoT can even be extended with Self-Consistency [12]. ReAct [13] interleaves Chain-of-Thought reasoning steps with calls to external APIs, whose results are processed by subsequent inference steps for generating the final response. Since its introduction, the ReAct "think, act, think, ... respond" pattern has become fairly common for tool-using agents. PAL [14] integrates LLMs with a Python interpreter to generate reasoning steps with Python code. Note that communication is one-way, with no option for feedback.

LLM agents can learn from their history of interactions with users and the external world. They are often implemented as a set of collaborating LLM modules. LLM agent components include an action/output generator (the "proposer") and a critic and/or learner (Reflexion [15]). LLM agents often employ some form of external memory system.

B. Retrieval Augmented Generation

LLMs often generate incorrect information, rely on outdated knowledge, or reach false conclusions through incorrect reasoning. These challenges are particularly evident in domain-specific or knowledge-intensive tasks which require up-to-date information or information not present in the training data. LLMs often confabulate facts ("hallucinate") in such situations. Retrieval Augmented Generation (RAG) [16] has emerged as mitigation to these issues by amending the user's input prompt with reliable, up-to-date data from reliable knowledge sources.

The RAG pipeline utilizes a knowledge base that can be searched by the LLM agents. First, raw data is extracted from PDF, HTML, Word, Markdown or even multimodal documentation. Then the material is cleaned and broken down into smaller segments called *chunks*. These chunks are subsequently encoded into a vector representations using an embedding model, and stored in a vector database. The choice of the embedding model matters, especially for languages other than English and for company-specific terminology: on a Hungarian corporate document set, retrieval quality varied substantially across eight multilingual embedding models [17]. The user's query is also converted into the same embedding vector representation which was used for indexing the reference documentation.

The naive RAG approach can be refined to serve specific use cases. Recursive Abstractive Processing for Tree-Organized Retrieval (RAPTOR) [18] implements a hierarchical, tree-based retrieval system. This improves Contextual Depth and enables efficient Multi-Step Reasoning by employing recursive summaries. Some other methods implement knowledge graphs [19] or combine RAG with agentic approaches that execute multiple retrievals [20].

C. Program Code Generation with LLMs

Besides processing natural language texts, LLMs can also deal with formal languages such as specifications, logics, and programming languages. LLMs can be used to generate, summarize, or modify programming languages and documentation. They can also translate from a natural language to source code (like Python or JavaScript), or modify existing code (e.g. add features, generate unit tests, or fix bugs) [21]–[26].

D. LLM Application Architectures

The typical LLM-based architecture is composed of a pretrained Large Language Model, external “tool” APIs, databases or vector stores for RAG, multiple specialized prompts or models, and external solvers or verifiers. Implementing the system requires a fair amount of coding for handling control flow, business logic, connection to cloud-based APIs, handle errors, etc. Various frameworks have been developed for building such systems. LlamaIndex (<https://llamaindex.ai>) and LangChain (<https://langchain.com>) are two of the most popular such frameworks. Originally developed for supporting RAG, both frameworks today help building complex agentic systems and work with third-party libraries and services.

LLM-based chatbots rely on Natural Language Understanding (NLU) and Natural Language Generation (NLG) support. LLMs in customer service provide personalized support, automated content generation, and sentiment analysis. Integrating LLMs into Customer Relationship Management (CRM) systems improves response relevance. The benefits of LLM-powered chatbots include scalability, consistent response quality, customization for brand alignment, and expenditure savings. Challenges still persist, such as addressing the risk of factually incorrect responses, non-compliance with data privacy laws. Handling out-of-domain queries requires hybrid models and fallback mechanisms.

The formal evaluation of LLMs improves their accuracy, safety, and reliability. **Factual correctness** is evaluated in question answering benchmarks and by measuring factuality in LLM-generated summaries. Core benchmarks are FEVER [27], FRANK [28], WikiFact [29], SQuAD [30]. Evaluation methods include FactScore [31], QAGS [32], BERTScore [33], and FactCC [34]. **Logical reasoning** is assessed through commonsense, mathematical (GSM8K [35]), logical (FOLIO [36], LogiQA [37]), and coding benchmarks. **Language Proficiency** evaluation assesses the model’s ability to generate fluent, coherent, and contextually appropriate language. This includes assessing grammar and style, context preservation, and adaptability across diverse prompts. Classic metrics like BLEU [38] and ROUGE [39] compare LLMs’ outputs with reference data. The Arena chatbot [40] is the primary platform for evaluating conversational abilities and interaction quality. **Alignment** ensures that content produced by LLMs is ethical, non-biased, and in line with human values. TrustGPT [41] among others measures bias, toxicity, and truthfulness of the model’s outputs. **LLMs as agents**. AgentBench [42] is a benchmark for evaluating the agency of

LLMs, assessing their decision-making, planning, and interactive capabilities. Reinforcement learning from human feedback (RLHF) is also used to address biases.

LLMs depend on large and high-quality datasets, which involves efficient data collection, preprocessing, and versioning. Ensuring privacy and compliance with regulations like GDPR or CCPA requires secure handling of sensitive information throughout the model lifecycle. LLMOps frameworks streamline model lifecycle management, ensuring efficient deployment and continuous improvement. LLMOps builds on MLOps practice that is already established in smart manufacturing, where automated pipelines manage data generation, training, and deployment of models on the factory floor [43]. Model versioning tracks changes to code, model architectures, training data, and parameters across different iterations. This helps organizations collaborate effectively, speeds up experimentation, and ensures compliance with regulatory or auditing requirements. Version control also supports reproducibility for debugging, analysis, or further development of the models.

Security practices like regular vulnerability assessments, coupled with redundancy measures, ensure model availability and robustness. By integrating these practices, organizations can overcome the complexities of LLM deployment and optimization, achieving scalable and reliable AI systems [44], [45].

LLMs require significant computational resources for training and inference, often necessitating specialized hardware like GPUs or TPUs. This results in both high upfront costs and high operational expenditure, for organizations running models at scale. For real-time applications like chatbots or recommendation systems managing latency and throughput can be a complex task [46]. In cloud-only systems users may be concerned with transmitting personal data. A shift toward hybrid/local deployment is gaining traction, when some or all processing is done locally or (in and edge scenario) close to the user. On-device processing often requires techniques like model pruning and quantization to adapt to limited local resources. NVIDIA TensorRT or ONNX Runtime for deep learning optimization can improve performance. In the long term, managing LLM costs requires a balance between model size, infrastructure investment, and optimized deployment strategies to ensure sustainability without sacrificing performance [47], [48].

E. LLM Applications in Industry

Several recent surveys map the use of LLMs in manufacturing. Ma et al. [49] review LLM applications across intelligent manufacturing and position LLMs as a key enabling technology of Industry 5.0. Zhang et al. [50] propose an LLM-enabled manufacturing framework spanning design, production, and service. Li et al. [51] evaluate the capabilities of general-purpose multimodal models such as GPT-4V on manufacturing tasks ranging from product design to quality control and supply chain management. Sadeqi Bajestani et al. [52] focus on human-in-the-loop integration and argue for a hybrid intelligence model in which LLMs automate tasks while human experts remain responsible for contextual

accuracy and quality assurance – a view closely aligned with the human-centric pillar of Industry 5.0.

The most mature line of application work is the generation of industrial control code. Koziol et al. [53] tested GPT-4 on 100 prompts in ten categories and found that it frequently produces syntactically correct IEC 61131-3 Structured Text, but that the results still require validation by experienced control engineers. LLM4PLC [54] addresses this by placing the LLM in a loop with grammar checkers, compilers, and formal verification tools, which raised the generation success rate from 47% to 72% and the expert-rated code quality from 3.0 to 7.2 out of 10. In robotics, Code as Policies [55] showed that code-writing LLMs can compose perception and control APIs into executable robot policies from natural-language commands – the same principle that our robot programming use case builds on.

LLM agents are also being used to orchestrate production systems. Xia et al. [56] connect LLM agents to a modular production system through its digital twin and service interfaces, enabling the agents to interpret system information and plan and execute production tasks. Lim et al. [57] extend multi-agent manufacturing systems with LLMs so that agents can interpret human instructions in natural language and coordinate tasks such as G-code allocation among machines. For industrial asset operations and maintenance, AssetOps-Bench [58] provides a benchmark of 141 scenarios in which domain-specific agents for IoT data, failure modes, time-series forecasting, and work orders must collaborate.

Domain adaptation is a recurring theme in engineering assistants. ChipNeMo [59] adapts open LLMs to semiconductor design through domain-specific tokenization, continued pre-training, instruction alignment, and domain-adapted retrieval; its largest variant outperformed GPT-4 on an engineering assistant chatbot and on EDA script generation. Field evidence on the productivity effect of such assistants is still scarce. A notable exception is the study of Brynjolfsson et al. [60], who measured a 15% average increase in issues resolved per hour among more than 5,000 customer support agents using a generative AI assistant, with the largest gains for less experienced workers. On the industrial vendor side, copilots are moving from question answering towards agents that execute complete engineering workflows, such as those announced for the Siemens Industrial Copilot ecosystem [61].

Most of these works examine a single task or a single lifecycle phase, and only a few report results from industrial deployment. The present paper complements them with a cross-case view of LLM applications across engineering, operations, after-sales support, compliance, and knowledge management within one industrial consortium.

III. METHODOLOGY

In this paper we adopt a use-case-driven qualitative methodology to investigate how LLM-based industrial applications contribute to the goals of Industry 5.0. Rather than evaluating a single technical solution, we consider a heterogeneous portfolio of application scenarios that span system-of-systems engineering, customer support, regulatory self-assessment, product

engineering, factual correctness checking, robot programming, ontology and knowledge-graph generation, and AI-gym-based evaluation. All of them were designed and implemented by partners of the AIMS5.0 project, and each is reported in Section IV by the team that developed it. The paper is therefore a multi-case experience report: it describes the cases along common dimensions, states their maturity explicitly, and interprets them against the objectives of Industry 5.0.

A. Methodological premise

The methodological premise is that the value of LLMs in Industry 5.0 cannot be understood solely through task-level performance metrics such as latency, accuracy, or success rate. LLM integration affects industrial work more deeply than a single task: it changes how engineers, operators, and service staff access knowledge, how they interact with machines and software systems, and how responsibility is shared between humans and automated components. The contribution of LLMs must therefore be interpreted in relation to broader socio-technical objectives.

For this reason, the analysis is framed around the three widely recognized Industry 5.0 pillars: human-centricity, sustainability, and resilience. In this work, *human-centricity* refers to the extent to which a use case augments human capabilities, reduces cognitive burden, supports explainability, and preserves meaningful human oversight. *Sustainability* refers not only to direct environmental effects, but also to resource efficiency, waste reduction, knowledge reuse, and lifecycle optimization. *Resilience* refers to the capacity of a system to remain functional, adaptable, safe, and trustworthy under uncertainty, change, partial failure, or evolving regulatory and operational conditions.

B. Case selection and lifecycle coverage

The use cases were selected because they represent distinct stages of the industrial lifecycle, involve different stakeholder groups, and address different forms of cognitive, operational, and organizational complexity. Taken together, they provide a broad empirical basis for examining the role of LLMs beyond narrow automation objectives.

To organize the cases, we follow the lifecycle view of human-centered AI in smart manufacturing proposed by Martini et al. [62], which distinguishes the design, production, and maintenance phases (Figure 1). The design phase is addressed by the Arrowhead Copilots for system-of-systems engineering, the RAG chatbot in product engineering, the ontology and knowledge-graph tools, and the AI Act self-assessment chatbot, which supports compliance checks early in development. The production phase is addressed by robot programming, where generated control code allows a production line to be reconfigured quickly, and by the Arrowhead Management Copilot, which interacts with running services. The maintenance phase is addressed by the support and after-sales chatbot. Factual correctness checking and the AI-gym are cross-phase enablers: they validate the output of LLMs and the behavior of AI models before these are relied upon in any phase. Table ?? lists the lifecycle phase of each case.

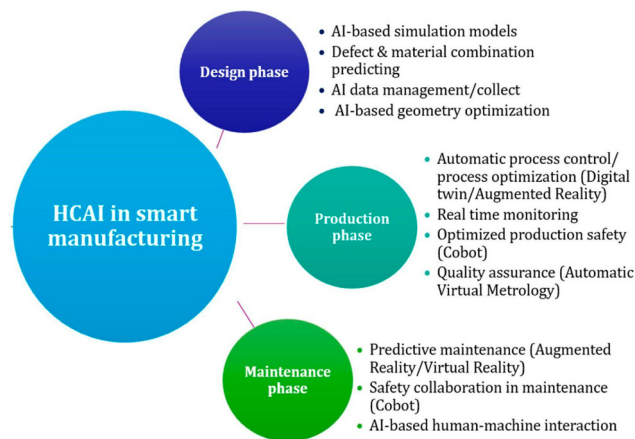


Fig. 1: Human-centered AI (HCAI) in smart manufacturing across the design, production, and maintenance phases. Reproduced from Martini et al. [62] (CC BY). The use cases of this paper are assigned to these phases in Table ??.

C. Analytical framework

The research design is a qualitative cross-case analysis. Each use case is treated as an individual unit of analysis, and all cases are assessed using a common analytical framework. The framework consists of five dimensions.

First, the *industrial context* is identified, including the lifecycle phase addressed by the use case, the primary stakeholder groups involved, and the type of industrial process or ecosystem in which the solution operates.

Second, the *LLM functional role* is characterized, such as retrieval-augmented assistance, code generation, planning and orchestration, compliance guidance, natural-language querying, semantic modeling support, or evaluation in simulated environments.

Third, the *human-system relationship* is analyzed, with particular attention to whether the LLM acts as an assistant, a co-pilot, an autonomous component, or a supervised recommendation engine.

Fourth, the *Industry 5.0 relevance* of the use case is assessed by mapping its expected effects to the three pillars.

Fifth, the *maturity and evidence* of the use case are recorded: how far the solution has been developed, what evidence is available, and which technical and organizational challenges were encountered during design or implementation.

D. Interpretation scheme for the Industry 5.0 pillars

We applied a structured interpretation scheme to make the mapping to the pillars work. A use case is considered to support human-centricity if it reduces the effort required to perform complex engineering or operational tasks, improves accessibility for non-expert users, supports better decision-making, or preserves the possibility of human intervention and accountability. A use case is considered to support sustainability if it contributes to shorter engineering cycles, reduced rework, lower material or energy waste, better resource allocation, more efficient support processes, improved traceability,

or longer useful lifetimes of products and systems through better knowledge access and maintainability. A use case is considered to support resilience if it improves adaptability to changing operating conditions, strengthens robustness against documentation gaps, errors, and uncertainties, enables safer or more reliable system behavior, supports compliance with legal and ethical constraints, or enhances the capability to validate and monitor industrial AI solutions before deployment.

The goal is not to force each use case into all three pillars equally, but to identify its dominant and secondary Industry 5.0 contributions. The methodology therefore allows asymmetric mappings: some use cases are primarily human-centric, others primarily resilience-oriented, while some have stronger sustainability implications through efficiency and optimization effects. This approach reflects the practical reality that Industry 5.0 alignment is often multidimensional but unevenly distributed across applications.

To improve rigor, our methodology distinguishes between direct and indirect links to Industry 5.0. Direct links occur when a use case explicitly addresses a pillar, for example through human-in-the-loop design, sustainability-oriented supply-chain transparency, or safety and compliance mechanisms. Indirect links occur when the primary technical function leads to second-order Industry 5.0 benefits, such as better documentation access leading to less rework, or improved semantic interoperability enabling more robust collaboration across organizations. This distinction prevents overclaiming while still recognizing broader lifecycle effects. It also helps articulate how LLM applications that may initially appear narrow or productivity-focused can still contribute to the strategic goals of Industry 5.0 when embedded into larger industrial workflows.

The mapping was agreed among the contributing authors. It expresses the intended contribution of each solution; the maturity levels described below show how far that contribution has been demonstrated so far.

E. Interpretation of the use cases

From this perspective, the Arrowhead Copilots use case is interpreted as a human-centric and resilience-oriented case, since it reduces the cognitive burden of SoS engineering while improving adaptivity in heterogeneous service environments. The support and after-sales chatbot is primarily human-centric, as it improves accessibility and continuity of service, but it also contributes to sustainability by reducing repetitive manual workload. The AI Act self-assessment chatbot is strongly linked to resilience, because it helps organizations cope with regulatory complexity and compliance risk, while also supporting human-centricity by making governance knowledge accessible to non-specialists. The product engineering RAG chatbot mainly addresses human-centricity and sustainability by improving access to engineering knowledge, reducing duplicated effort, and accelerating document-driven tasks. The factual correctness checking use case is treated as a resilience-enabling mechanism, because it increases trustworthiness in AI-supported query generation and reduces the risk of incorrect machine-mediated decisions. The robot programming use

case spans all three pillars: it is human-centric in reducing programming effort, sustainable in minimizing reconfiguration cost and downtime, and resilient through simulation-based validation and safer deployment of generated code. The ontology generation and knowledge-graph use case contributes to human-centricity by lowering semantic barriers, to sustainability through improved reuse and interoperability of industrial knowledge, and to resilience by enabling robust cross-organizational data integration. Finally, the AI-gym use case is interpreted primarily through the resilience lens, since it provides a controlled environment for safe testing, continuous refinement, and reduced deployment risk, while also indirectly supporting human-centric engineering practices. Table I summarizes this mapping.

F. Limitations of the approach

The case descriptions are written by the teams that developed the solutions, so they are subject to self-reporting bias. The cases use different metrics, and most of them are at the concept or PoC level. The findings should therefore be read as practical experience and design patterns, not as generalizable performance results.

G. Summary of the methodological approach

Hence, the proposed methodology provides a reusable interpretive framework for connecting industrial AI use cases to Industry 5.0. Its central assumption is that the relevance of LLMs in industrial settings should be evaluated not only by technical performance, but also by their capacity to augment people, reduce waste and inefficiency, and strengthen the robustness and adaptability of industrial systems. By applying a common analytical lens to a diverse set of use cases, the study develops a consolidated view of LLMs as enabling components of a more human-centered, sustainable, and resilient industrial future.

IV. APPLICATION AREAS FOR LLMs IN INDUSTRY 5.0

A. Arrowhead Copilots for System-of-Systems Engineering

System-of-systems (SoS) engineering in Industry 5.0 environments involves orchestrating distributed, heterogeneous, and autonomous subsystems, each with its own interfaces, constraints, and lifecycles. Engineers face difficulties in navigating service contracts, maintaining compliance with framework specifications, and designing complex orchestrations across distributed industrial components. Studying documentation, API contracts, and semantic service descriptions is time-consuming and error-prone, particularly when working under time constraints or with legacy systems.

The integration of LLM-based tools into the engineering workflows addresses these systemic problems, speeds up development, and improves productivity. AI agents can quickly search fragmented documentation and assist in interpreting complex service descriptions. Engineers can use natural language instructions, which are automatically translated to programmatic requests – saving the time and effort of learning different subsystem interfaces.

To showcase the power of LLM tools in the system-of-systems engineering process, we developed three agents for the Arrowhead SoS Orchestration Framework, each assisting the workflow in its own way (Figure 2). The framework itself continues to evolve; for example, recent work extends it with service-level synchronization for time-critical applications while preserving late binding and loose coupling [63]. The copilots therefore have to keep up with a growing set of services and documentation.

The *Arrowhead Expert* is a multi-modal chat agent, which uses RAG to quickly find relevant information in a pool of documents about a subsystem or component. The primary advantage of this agent is its ability to handle documents from diverse sources and of various types, such as technical documentation, scientific papers, and flowcharts. The uploaded documents are divided into smaller, semantically coherent segments, which are then converted into embedding vectors, in a way that more fragmented but relevant partitions can also be found effectively. The agent is multi-modal: it can process pictures, tables and figures as well. The responses are accurate, and can even contain relevant document sections or quotes if the user requests more insight.

The *Arrowhead Management Copilot* takes natural language queries in order to programmatically interact with different subsystems. It processes the API documentations of the available components, selects the relevant services, creates and executes the required API calls, and evaluates their results – according to the user's request. The tool was created to handle complex tasks and interact between multiple services, rather than just answer simple queries. To achieve this, we developed a multi-agentic planner engine, which divides the workflow into smaller sub-tasks, each one handled by a specific AI agent. The planner agent devises a plan on how to best achieve the user's goal. The plan is executed step-by-step; each intermediate result is evaluated, and the plan is reviewed if necessary – e.g. on service error or unexpected API call response. The multi-agentic approach allows the planner engine to make changes to its own plan during execution. This makes the tool highly autonomous, adaptive, and robust.

The *Arrowhead Design Copilot* tool helps programmers in software development, SoS orchestration, and the creation of SysML diagrams that integrate into the Arrowhead Framework services.

Using proper guardrail mechanisms, in the Arrowhead Expert we filter out user requests which are not related to the stored documents' domain. In addition to the generic profanity and sensitive content filters, implemented additional guardrails to make sure the queries refer to the use-case context. This has been challenging, because new uploads to the document pool or deleting existing documents may shift the use case domain. The Management Copilot, while being useful and adaptive, struggles interacting with poorly documented components. Some (especially legacy) services don't have proper documentation, do not conform to the documentation, or do not provide helpful error messages. Even if workarounds exist in some cases, such issues need to be considered when selecting the SoS services.

TABLE I
MAPPING OF THE ANALYZED LLM-BASED INDUSTRIAL USE CASES TO THE PILLARS OF INDUSTRY 5.0. PRIMARY CONTRIBUTIONS ARE MARKED WITH X, SECONDARY CONTRIBUTIONS WITH O. THE MARKS EXPRESS THE INTENDED CONTRIBUTION OF EACH SOLUTION.

Use case	Main LLM role	Human-centricity	Sustainability	Resilience	Main contribution to Industry 5.0
Arrowhead Copilots for SoS Engineering	RAG assistance, multi-agent planning, design support	X	o	X	Reduces engineering cognitive load, improves accessibility of complex SoS knowledge, and increases adaptability in heterogeneous service environments.
Chatbots for Support and After-Sales Services	RAG chatbot, ticket handling, human handover	X	o	o	Improves user experience and support accessibility, reduces repetitive manual workload, and preserves continuity through human escalation.
AI Act Self-Assessment Chatbot	Compliance-oriented RAG guidance and reporting	o	o	X	Strengthens governance, legal robustness, and trustworthy deployment of industrial AI by helping developers navigate complex regulations and standards.
RAG Chatbots in Product Engineering	Enterprise knowledge retrieval and grounded answering	X	X	o	Improves access to engineering knowledge, reduces duplicated effort and search time, and supports more efficient engineering workflows.
Factual Correctness Checking	LLM output verification, semantic ranking, NL2SPARQL validation	o	o	X	Increases trustworthiness of AI-assisted querying and reduces the risk of incorrect machine-generated decisions in industrial knowledge access.
Robot Programming	Natural-language-to-code generation with simulation-based validation	X	X	X	Lowers the entry barrier for robot programming, reduces downtime during task changes, and improves safe adaptation through simulation-backed verification.
Ontology Generation and Knowledge Graphs	Ontology support, NL querying, semantic integration	X	X	X	Reduces semantic complexity for users, improves reuse and interoperability of industrial knowledge, and supports robust multi-stakeholder data integration.
Testing LLMs with Arrowhead-compliant AI-gym	Safe evaluation, iterative model refinement, simulation-based testing	o	o	X	Enhances robustness and safe deployment of industrial AI by enabling controlled testing in realistic but low-risk environments.

B. Chatbots for Support and After-Sales Services

Support and after-sales services frequently face the challenge of delivering timely assistance to a diverse customer base, often across multiple languages and for non-critical issues that could be addressed through self-service. These inefficiencies lead to unnecessary workload for human agents and decreased customer satisfaction. A significant portion of customer requests are repetitive and involve information that is already available in company resources, such as manuals or frequently asked questions. However, these documents are often underutilized due to poor accessibility or inadequate formatting for end-user consumption.

To address these issues, the integration of Large Language Models (LLMs) into customer support ecosystems offers a powerful solution. By embedding LLMs within a Retrieval-Augmented Generation (RAG) framework, companies can design chatbots capable of providing intelligent, real-time support based on both structured and unstructured internal data. The cognitive architecture of such a system, depicted in Figure 3, includes three core components: dynamic interaction with existing ticket management systems, retrieval-based access to enterprise knowledge bases, and a live chat interface that facilitates smooth handovers to human agents when necessary. The chatbot interacts with ticketing platforms via APIs to create, update, and categorize tickets based on user input. At the same time, it queries internal documentation – such as technical manuals, past support cases, and FAQs – via retrieval pipelines to generate accurate, context-aware responses. When a query exceeds the chatbot's confidence threshold or requires

expert intervention, it transfers the conversation to a human agent along with the full context to ensure continuity and efficiency.

During implementation, several lessons emerged. A key insight was the importance of user-centered conversational design. Using natural language aligned with the company's tone improves the perceived quality of the interaction. Offering guided prompts and quick replies helps streamline the user journey, especially for frequent tasks. Presenting information dynamically – such as summarizing ticket status or displaying progress indicators – also contributes to better transparency and user trust. Enabling a seamless transition to human agents while preserving the conversational context greatly enhances both customer satisfaction and resolution efficiency.

Despite these advantages, the deployment of LLM-based support chatbots comes with notable limitations. Preparing knowledge assets for retrieval poses a significant challenge. Manuals and technical documents are often multilingual, inconsistently formatted, and include non-textual elements like images and diagrams, which require sophisticated multimodal processing and custom extraction pipelines. Ensuring data privacy and removing sensitive information adds another layer of complexity. Evaluating chatbot performance is also non-trivial. Some metrics, such as response latency and ticket deflection, can be tracked automatically. However, evaluating the accuracy and contextual relevance of the chatbot's answers necessitates the early and ongoing involvement of domain experts. Another critical concern is the risk of confabulated or inappropriate responses. To mitigate this, it is essential to

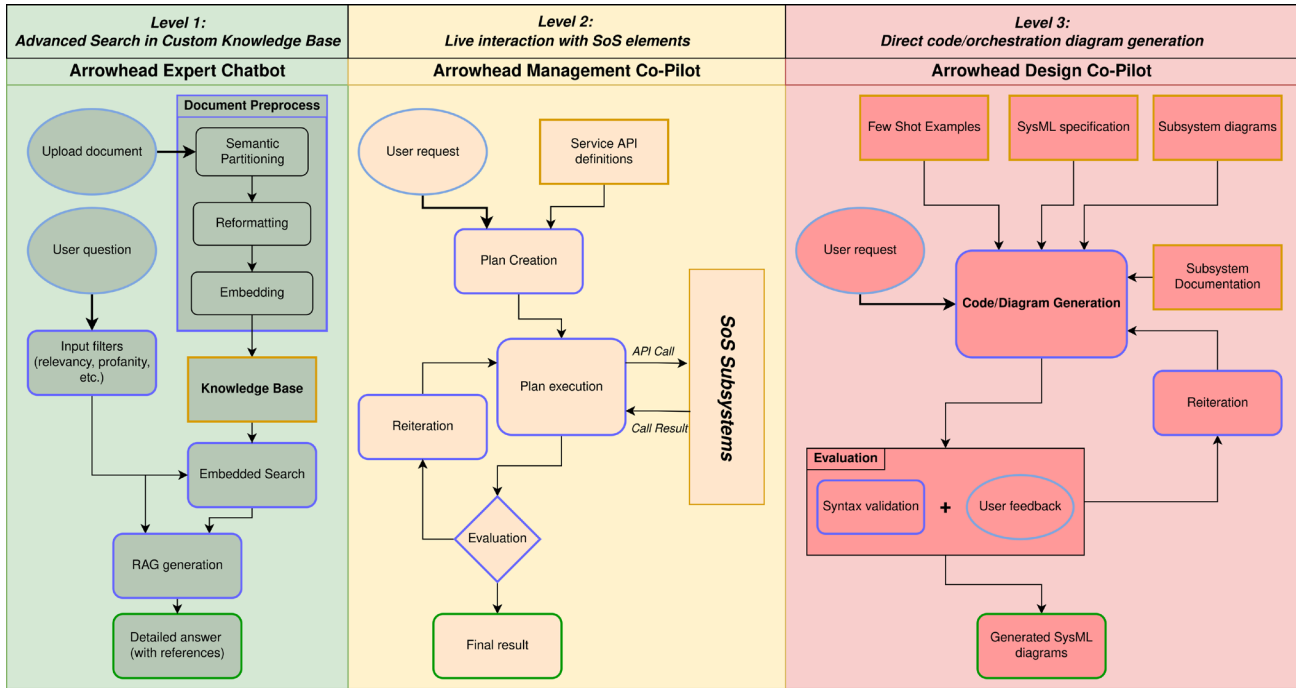


Fig. 2: Structure and workflow of Arrowhead Copilots: Expert Chatbot, Management Copilot, Design Copilot

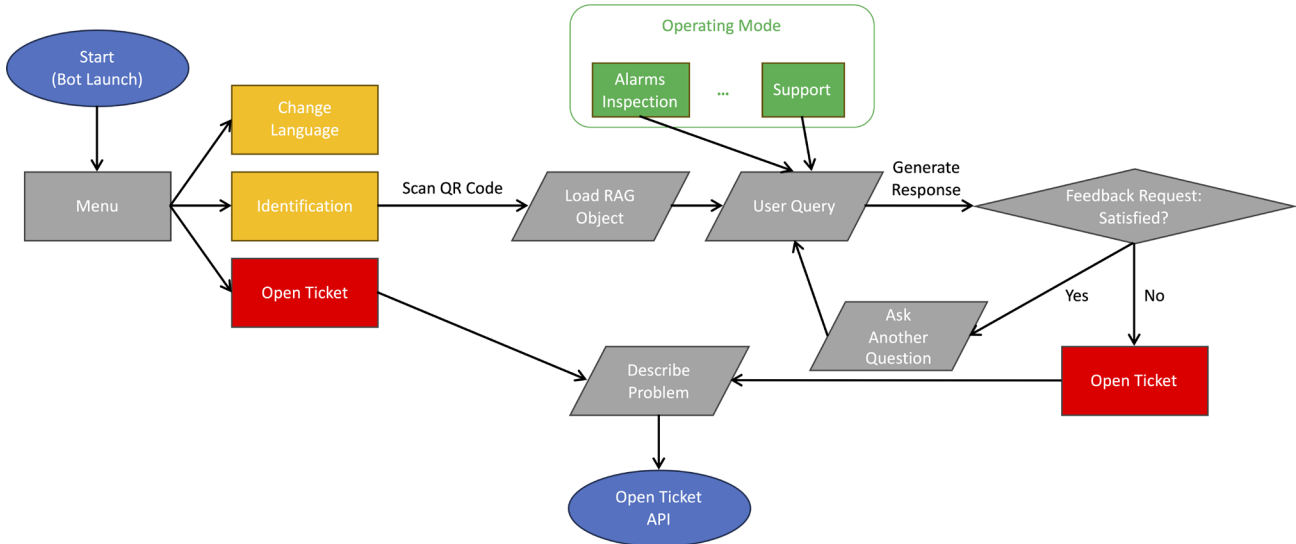


Fig. 3: A schematic description of the chatbot flow for Support and After-Sales Services

implement strict controls based on user identity and role. For instance, users of industrial equipment should never receive instructions to carry out expert-level maintenance procedures which they are not authorized or trained to perform.

In conclusion, the use of LLMs in customer support represents a leap forward in operational efficiency and user experience. By integrating retrieval-augmented reasoning into enterprise systems, LLM-powered chatbots can handle a broad spectrum of support requests autonomously. However, the successful adoption of such systems hinges on careful content preparation, ethical safeguards, and continuous collaboration

with human experts throughout the system lifecycle.

C. AIMS5.0 Chatbot to support self-assessment with the AI Act

The rapid evolution of artificial intelligence technologies enabled transformative applications across industries. These new services also introduced ethics, legal compliance, and governance challenges. As global and regional regulations gain traction – such as the European Union’s AI Act [64] –, developers and publishers of industrial AI systems face increasing complexity in navigating diverse guidelines, standards, and

legal frameworks. This complexity is compounded by the need to ensure ethical soundness, regulatory adherence, and conformance to national and international standards. Although there are tools based on Large Language Models that help users navigate these documents, such as ChatGPT, or tools provided by standardization organizations, these tools are either fundamental inquiry tools or very generic and do not cover copyrighted standards.

To address this challenge, we are building an innovative AI-driven self-assessment tool – a chatbot designed to assist AI developers and users in evaluating the compliance of their applications with ethical guidelines, legal regulations, and industry standards. Leveraging a Retrieval-Augmented Generation (RAG) framework – which combines an LLM with an external knowledge retrieval system – and hosted on cloud infrastructure, the chatbot provides real-time, context-aware guidance, allowing users to assess their AI systems efficiently. This solution not only encourages a consistent approach to ethics and verifies compliance, but also reduces the time and expertise required to navigate complex regulatory landscapes.

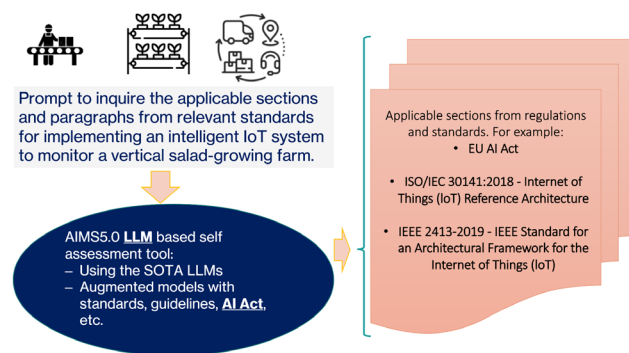


Fig. 4: A schematic description of the self-assessment tool

Integrating AI technology into a Retrieval-Augmented Generation system, we can develop a chatbot that helps application developers check if their work follows legal and ethical rules. This chatbot works as a self-assessment tool that stores important AI regulations, standards, and ethical guidelines in a local database. When developers chat with the system and answer specific questions about their AI project, the chatbot searches through this database to find relevant information and creates a personalized report showing which regulations, standards, and guidelines apply to their situation. This allows developers to check their applications early in the development process, ensuring they meet compliance requirements, safety standards, and ethical guidelines, which builds user trust and helps avoid regulatory violations. The system includes key documents such as the EU AI Act, Digital Services Act (DSA), Digital Markets Act (DMA), Transparency Standard, Bias Standard, HLEG guidelines on trustworthy AI, and various domain-related and technical standards. Figure 4 illustrates the schematic arrangement of the self-assessment tool.

D. RAG Chatbots in Product Engineering

In Signify (formerly Philips Lighting) the integration of a Retrieval-Augmented Generation chatbot in product engineer-

ing aims to improve productivity and quality by providing easy access to critical engineering documents. The SharePoint server hosts product compliance guidelines, manufacturing processes, design rules, component information, and design tool documentation. The current proof-of-concept is a “naive” RAG solution built on Azure components as shown in Figure 5. The initial RAG system combines the Azure OpenAI On Your Data service with Azure AI Search, and gets updated when new frameworks are available under Azure. The Engineering Portal documents have been indexed with vector embeddings in Azure AI Search. The Azure OpenAI On Your Data service takes the user question with the conversation history and uses an LLM to determine the user intent. The intent is used to query Azure AI Search to produce relevant context, which is then used by Azure OpenAI On Your Data service to make another LLM completion call to generate the response. While this Azure-based approach leverages a robust cloud ecosystem, its implementation has revealed several key challenges:

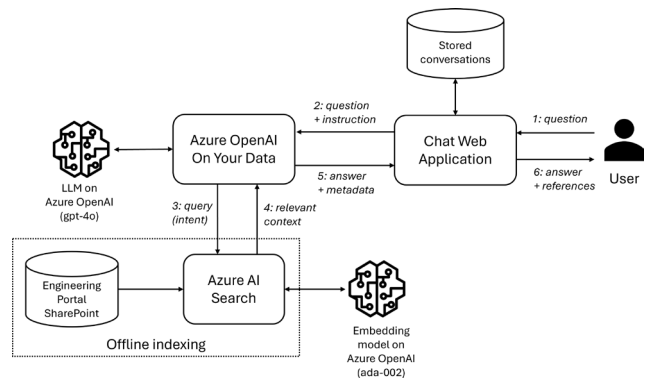


Fig. 5: Azure based architecture of Engineering Portal Chat.

Deployment and Data Security: The solution is deployed entirely within Azure. A SaaS solution may offer ease of deployment but raises concerns about data leakage, especially when sensitive engineering documents are processed by cloud-based language models. Ensuring data confidentiality is critical, as exposure of proprietary information could have severe implications for the company.

Fine-Grained Access Control: The RAG system must enforce document-level permissions, ensuring that users can only access information they are authorized to view. In a “naive” RAG implementation unauthorized documents can be filtered out in the retrieval step. In more sophisticated architectures, like graph-based RAG or hierarchical models such as RAPTOR, security enforcement is more challenging to implement properly.

Integrating Distributed and Multi-Modal Knowledge: The Engineering Portal documents contain links to web pages and documents stored in other systems, comprising different data modalities such as text documents, tables, graphs, and presentations. Aggregating this distributed and multi-modal data into a unified system is an open challenge and likely demands advanced RAG solutions.

Maintaining Data Currency: Engineering documents are

being updated perpetually. Our RAG implementation must regularly check for changes in the SharePoint content and re-index it, so that users always receive the most up-to-date technical references. When improving quality with more advanced RAG solutions, regular and incremental updates should be accounted for.

Ensuring Response Quality and Accuracy: All steps of the RAG process – indexing, retrieval and generation – contribute to response quality. The system prompt's wording also plays a crucial role in achieving meaningful results. In our implementation, the prompt defines the knowledge base scope and establishes answering rules that constrain the LLM to grounded responses while preventing confabulations such as fabricating formulas. Consistently delivering factually accurate and detailed responses remains a significant challenge, particularly when dealing with technical specifications or complex design rules.

Enhancing User Experience: A smooth user experience is essential, because many engineers may be unfamiliar with AI chatbot interactions. The customized system prompt serves two purposes: it instructs the LLM on content boundaries, and informs users about the chatbot's scope and limitations. The users can provide feedback about the chatbot's performance, which is stored alongside the conversations. Analyzing feedback and chatbot usage patterns is essential for improvement, but maintaining privacy remains our top priority.

This exploration of RAG chatbot integration in product engineering has highlighted fundamental challenges in industrial LLM applications. Resolving these challenges is essential for realizing the complete potential of LLMs in industry.

E. Factual Correctness Checking

Artificial intelligence-generated content can substantially improve decision-making across various industries. Integrating large language models with knowledge graphs enables natural language (NL) querying, allowing non-experts to access complex data without needing to master SPARQL. However, because LLMs can generate non-factual or inaccurate outputs, it is critical to ensure the correctness of the translated queries.

The translation of NL to SPARQL (NL2SPARQL) poses multiple challenges. Proprietary models such as GPT-4 [65] often cannot be used in industrial settings, because queries and schema information would have to leave the organization. Open-weight models that can be deployed on premises have narrowed the gap to proprietary models considerably: DeepSeek-V3 [66], DeepSeek-R1 [67], Qwen3 [68], and gpt-oss [69] report performance comparable to leading closed models on reasoning and coding benchmarks, and gpt-oss-120b has been used for SPARQL generation over a corporate knowledge graph [70]. General capability, however, does not guarantee correct queries over a private schema: the model has to use the classes and properties of the target ontology correctly, and the NL-SPARQL pairs needed to fine-tune or evaluate a model for a specific knowledge graph are hard to obtain. For this reason, the first TEXT2SPARQL challenge evaluated systems on a corporate knowledge graph in addition to DBpedia [71]. Methods that verify the semantic accuracy of a generated query before it is executed remain scarce.

To overcome these issues, we propose an approach in which an open-source language model generates multiple candidate SPARQL queries which are then ranked by their semantic similarity to the original NL query. Each SPARQL candidate is encoded into a dense representation and compared with the dense representation of the NL question using a semantic similarity model. For encoding, BERT [72] can be employed, fine-tuned with adapters on both NL questions and SPARQL queries. A multilayer perceptron (MLP) can be trained for similarity scoring. Training this semantic similarity model requires a parallel dataset of NL questions and SPARQL queries [73].

In the processing flow (Figure 6a), from the initial NL question the LLM generates k SPARQL candidates, which are ranked by their semantic similarity to the NL question. The highest-ranking candidate is then used to query the knowledge graph. A candidate is deemed correct if its output matches that of an expert-crafted query, as illustrated in Figure 6b. The collection of expert-crafted queries is incorporated as part of the use-case. This pipeline generates multiple candidates by prompt-tuning of LLMs. As there is no need to train models from scratch, it is sufficient to fine-tune a smaller encoder and train only a small-scale MLP. We assume that generating multiple candidates increases the likelihood of producing a correct symbolic representation. This builds trust in LLM-generated queries, and promotes the adoption of AI solutions in industrial processes.

While the conceptual framework and system architecture have been fully specified, the proposed pipeline has not yet undergone empirical validation. A comprehensive evaluation—including ablation studies on benchmark datasets such as LC-QuAD 2.0 and domain-specific knowledge graphs—is planned for the next phase of this research. These forthcoming experiments will measure (i) the recall of correct SPARQL candidates generated by the LLM, (ii) the precision of the semantic-similarity ranker, and (iii) overall question-answering accuracy compared with current NL2SPARQL baselines. We expect that demonstrating robust performance across these metrics will provide clear guidelines for deployment in production settings. Results from these studies, together with an open-source reference implementation, will be reported in a subsequent publication.

F. Robot Programming

Industrial robots play an important role in modern manufacturing, contributing to precise and efficient automation of production processes. However, it is still difficult to provide detailed task specifications and implement them as robot control programs. Unpredictably positioned objects and variability of motion in the robot's environment are particularly challenging, often requiring integration of machine vision and real-time motion planning into the robot control system. These factors slow down the adoption of robots in industrial automation settings.

Generative AI holds the promise of generating control code from informal, high-level task descriptions. Trained on already available program samples, large language models

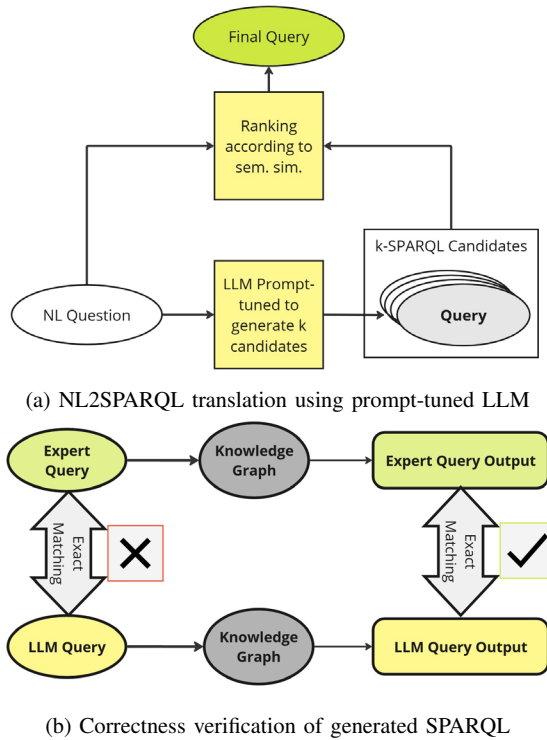


Fig. 6: Overview of NL2SPARQL translation and verification pipeline.

can translate natural language instructions into robot control source code. This reduces the reliance on robotics specialists, democratizes industrial robot programming, and cuts the time required for coding or production line reconfiguration. But applying generative AI introduces its own technical challenges. It is not trivial to ensure that the generated code invokes the machine vision and robot control APIs correctly. Robust verification mechanisms should be in place to rule out unintended or unsafe robot operation and guarantee successful task completion across a range of component positions and motion variations on the production line.

The focus of the AIMS5.0 project's Use Case 6 is to develop a robotic feeding system for a beverage bottling line. Our objective is to quickly switch tasks or introduce new tasks in the processing workflow, with minimal downtime. Tasks are defined by bottle kinds (differing in shape, size, weight, material, color); by bottle carrier-block types; and whether the bottles are piled randomly or neatly layered in the boxes delivered to the production line. The flow of carrier-blocks fluctuates in concordance with the complexity of the processing steps downstream on the production line, and is thus irregular and unpredictable at the bottle insertion station.

For our purposes, the key performance indicators are reliability, processing speed/throughput, safety, and reconfigurability. The entire pick-and-place pipeline (including computer vision, grasp pose estimation, motion planning, and dynamic control) should be fast enough to maintain the current throughput of 40–70 bottles per minute. Safety considerations are

critical – the risk of damage to health or equipment should be minimized to the extent engineering permits. Within these constraints the goal is to minimize the time, effort, and expertise required to switch from one task to another or to introduce a new one.

Figure 7 shows the high-level architecture of the code generation system. The LLM produces two kinds of source code: configuration code for task simulation, and robot control scripts. The task configuration code serves two functions: initializes randomized task simulations, and monitors task progress within the simulation – in order to provide feedback on task success. The robot control scripts call machine vision, motion planning, and robot control components which perform the actual task; such components are increasingly learned models themselves, for example transformer-based path planners [74].

The two kinds of source code are generated independently for multiple reasons: (1) This makes it easier to separate code targeting the simulation environment from code which interacts with the elevated privileges of real-world APIs; (2) the potential bugs in the generated code can be vetted and fixed independently; and (3) the proven versions are saved into distinct repositories of in-context examples for future code generation queries.

The system is intended to be adaptable to applications beyond the specific bottling use case described above. However, two key design assumptions limit the system's scope of applicability: (1) We assume a "closed world" scenario: all encountered object types are defined beforehand (with the option to add new types as needed). The physics models and all objects' spatial characteristics are known in advance. This is a reasonable assumption for the use case's stationary robots in the current manufacturing setting. (2) Rather than relying on large Language+Vision [75]–[77] or Vision+Action models [78], [79] to make inferences about object locations and orientations, we decide to invoke fine-tuned perception and grasp-estimation models from the robot control scripts. The limited-scope models are much smaller and faster, and thus can be run locally at the robot – rather than communicating with an inference server at a remote location. We also found these smaller models more predictable.

Some of the KPIs are directly incorporated into the code generation system. For instance, the generated robot control scripts are automatically evaluated in simulation for reliability, by executing them in multiple randomized simulation instances. Code generation contributes to speed and safety at the same time: vetted and approved scripts are handled as stand-alone, deterministic code modules; they can be inspected and deployed independently of the process that generated them.

The code generation system is evaluated using a manually curated, small set of example tasks and a fixed budget of generation attempts from natural language task descriptions. The results are derived from the following statistics: *Simulation Generation Success Rate*: how often the system generates usable simulation configurations; *Simulation Generation Partial Success Rate*: how often the system generates a simulation configuration that can be fixed with low effort by human

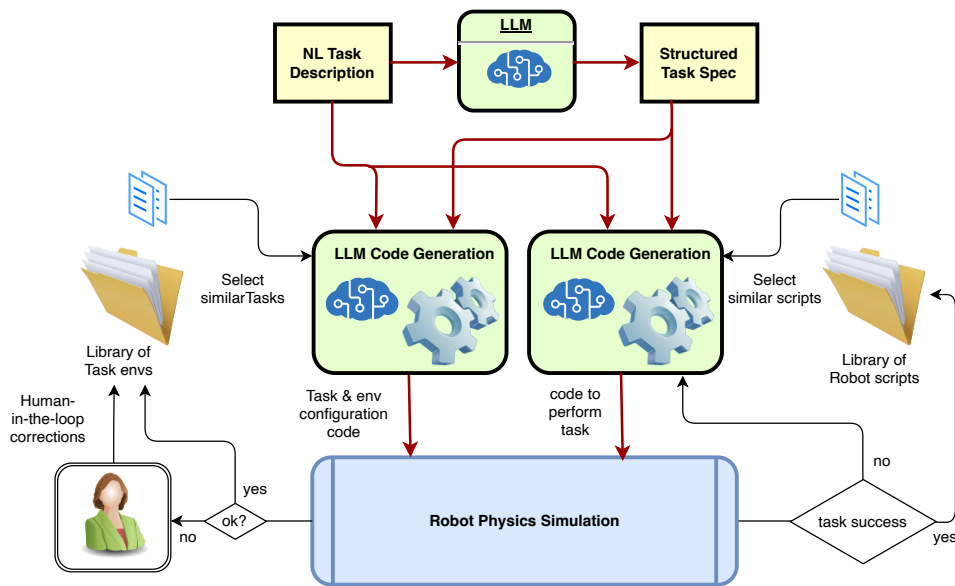


Fig. 7: From NL description, an LLM generates two kinds of scripts: simulation configuration, and high level robot control

intervention (cf.: human-in-the-loop effort/time); and *Script Generation Success Rate*: how often the system generates a robot control script that succeeds across a number of randomized simulation instances.

These metrics vary depending on the code-generation LLMs employed (together with the quality and number of few-shot examples for prompting). The metrics also depend on the sophistication of the physical simulation model, and thus on the generated code. All of these technologies are evolving rapidly, so detailed results at this stage would not be particularly informative. However, an initial proof of concept using GPT-4o as the code generation model for two basic tasks, along with a highly abstracted robot simulation in the *pybullet* environment, demonstrated that the system can generate viable robot control scripts. The results were deemed promising enough to validate the approach, and efforts are currently under way to implement a higher fidelity simulation environment incorporating more realistic robot perception and control primitives. A higher-fidelity simulation does not remove the gap between simulated and real perception; techniques such as visual domain randomization combined with real images [80] can make behavior validated in simulation more robust on the physical line. By outlining the key challenges, detailing the design choices and methods, and presenting an initial proof of concept, this use case aims to offer insights into how code generation AI models could broaden the range of manufacturing environments where industrial robots can economically be deployed.

G. Ontology Generation and Knowledge Graphs

The semiconductor industry requires integration of data generated by interconnected manufacturing processes, supply chains, and sustainability initiatives. Fragmented data

sources, inconsistent data formats, and poor interoperability among systems complicate this process. These issues hinder effective decision-making in core areas such as supply chain management, manufacturing optimization, and environmental impact tracking. For example, supply chain contributors often struggle to trace components and materials due to incompatible data structures. Similarly, isolated silos prevent manufacturing processes from getting real-time data insights. Addressing these challenges requires a common semantic representation, standardized knowledge-sharing mechanisms, and tools to enable efficient collaboration on data-intensive tasks.

The Semantic Web extends traditional web technologies by embedding machine-readable meaning into data through ontologies and knowledge graphs [81]. Ontologies serve as formal vocabularies, defining concepts and relationships in a way that both machines and humans can interpret. Knowledge graphs represent these ontologies as networks of interconnected industrial knowledge. The ontologies' semantic foundation enables systems to reason about complex manufacturing processes, supply chain relationships, and human-machine interactions.

The AIMS5.0 Open Access Platform represents a practical implementation of Semantic Web principles in industrial settings. The platform architecture is specifically tailored to modern semiconductor manufacturing processes and their extended supply chains, where data heterogeneity and interdependence create further operational complexities.

The platform's architecture (illustrated in Figure 8) allows the semantic integration of multiple datasets and ensures secure data exchange. Specific tools help stakeholders to map their data into knowledge graphs for aggregated data analysis and semantic reasoning. Large language models and AI-driven enhancements promote intuitive interactions with

data, reducing operational complexity and users' cognitive overload. This feature is particularly helpful for users who are unfamiliar with semantic technologies.

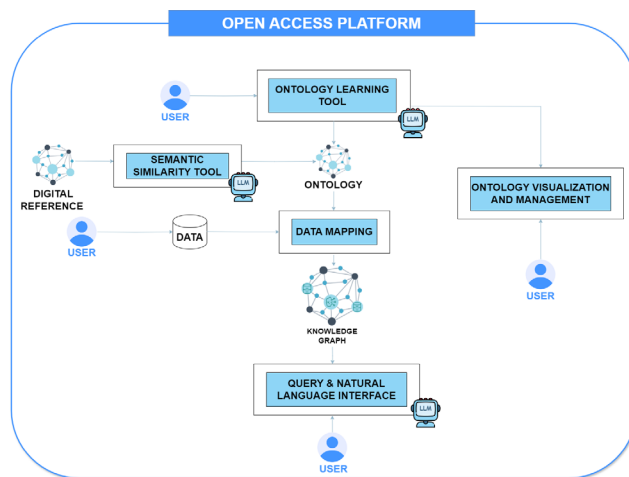


Fig. 8: Open Access Platform architecture

The AIMS5.0 platform architecture consists of two primary functional areas: a knowledge sharing infrastructure and a secure data transfer framework. In both areas LLM-based AI tools help reduce technical complexity, streamline workflows, and enable intuitive interaction with data.

The knowledge sharing infrastructure enables the creation, alignment, and management of semantic models through ontologies and knowledge graphs. Users can create domain-specific ontologies tailored to their datasets, even without prior expertise in semantic modeling. Tools like Chowlk assist in automatically converting data schemas into formal ontology files, simplifying the traditionally complex process of ontology creation. We added LLM-based tools for schema extraction from unstructured data, further reducing users' cognitive burden, and increasing accessibility for non-experts.

To ensure interoperability, user-created ontologies are aligned with the *Digital Reference* model [82], shown in Figure 9, which provides a comprehensive semantic representation of the semiconductor supply chain – comprising sub-ontologies for products, sensors, organizations, and business processes. This alignment minimizes redundancy, promotes reuse, and ensures seamless data exchange across stakeholders.

After ontology creation or alignment, datasets are semantically mapped to the data models. Visual mapping tools, such as OpenRefine, offer an intuitive graphical interface for users with limited technical knowledge. Expert users can rely on flexible, traditional mapping tools for greater control. Following semantic mapping, knowledge graphs are generated to represent the relationships among data entities, enabling semantic reasoning and analysis. Knowledge graphs are stored in graph databases, which support efficient querying through SPARQL and Cypher endpoints.

Large language models simplify user interactions with the knowledge graphs. For example, users can now pose questions in natural language, which are translated into SPARQL or Cypher queries. This feature allows access to semantic

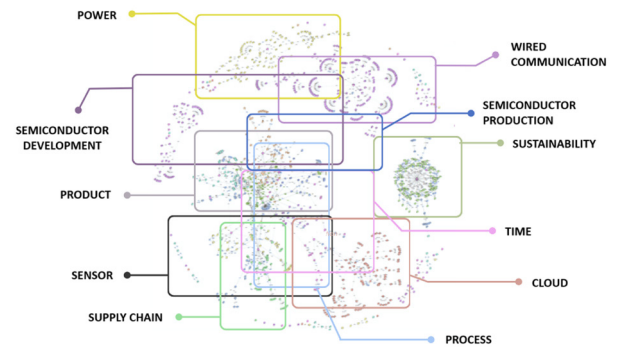


Fig. 9: Digital Reference Model of semantic representations

technologies for non-technical users and a wider range of stakeholders. By lowering technical barriers, these enhancements reduce cognitive overload and speed up the adoption of semantic, data-driven approaches.

The secure data transfer framework complements the knowledge sharing infrastructure by ensuring encrypted and controlled access to sensitive industrial data. Semantic integration across distributed datasets allows for collaborative analysis of complex ecosystems, such as interconnected supply chains. The framework also ensures that stakeholders can securely share insights derived from their knowledge graphs, promoting trust and collaboration.

The development of the AIMS5.0 platform has highlighted the importance of simplifying workflows for users. Early efforts show that LLM-driven enhancements, such as natural language querying and automated ontology creation, can greatly reduce cognitive overload and improve accessibility for non-technical stakeholders. Semantic integration, particularly through alignment with the *Digital Reference* model, highlights the value of interoperability and collaboration across fragmented datasets. Intuitive tools for mapping and visualizing knowledge graphs are proving critical for encouraging adoption and enabling data-driven decision-making.

However, several challenges remain. Ontology creation still demands considerable domain expertise and effort, which can slow down progress. Aligning legacy systems with the semantic framework is difficult, especially when data is incomplete or inconsistent. Some users, particularly those with limited technical expertise, may struggle to grasp semantic modeling concepts and interact with knowledge graphs. While LLM-based natural language querying addresses these issues, its effectiveness has not yet been fully validated. As industrial requirements evolve, scalability and long-term maintenance of ontologies and knowledge graphs become another key focus area.

Large language models have significant potential to address many of the platform's shortcomings. In ontology creation, LLMs can automate schema extraction from unstructured datasets, reducing the dependency on domain experts and accelerating the modeling process [83]. LLMs can improve ontology alignment by identifying semantic similarities between concepts, enabling seamless integration of multiple

ontologies [84].

LLM-driven enhancements can further simplify querying by translating natural language questions into SPARQL or Cypher queries, making semantic tools more accessible to non-technical users. Beyond querying, LLMs can support knowledge graph population through tasks such as entity recognition and dependency parsing, streamlining the integration of new datasets into the platform.

The AIMS5.0 Open Access Platform exemplifies the transformative potential of Semantic Web and AI technologies in addressing challenges of the semiconductor industry. By enabling semantic integration, knowledge sharing, and advanced reasoning, the platform has improved supply chain transparency, optimized manufacturing processes, and supported sustainability efforts. With the integration of LLM-driven enhancements, the platform is poised to further reduce complexity, improve usability, and drive growth and sustainability across the industry.

H. Testing LLMs with Eclipse Arrowhead-compliant AI-gym

Testing Large Language Models (LLMs) presents unique challenges in evaluating their robustness, ethics, biases, trustworthiness, and ability to adapt to dynamic real-world scenarios. Traditional testing methods for LLMs, such as static benchmarks and isolated evaluation metrics [85], have notable limitations. Chang et al. [86], highlight that current LLMs have limited capabilities in reasoning and robustness. They also underscore the pressing need for modern evaluation systems, so that the models' capabilities and constraints can be assessed more accurately. This calls for more robust evaluation frameworks that test LLMs in interactive, diverse, and evolving settings. The AI-gym concept offers a promising solution by providing simulated, interactive environments where LLMs can be tested across diverse tasks and evolving contexts. These environments allow for systematic benchmarking, continuous feedback loops, and safe exploration of complex scenarios, enabling developers to better assess and improve LLM performance in controlled yet flexible settings.

In this section we present an AI-gym implementation – a controlled environment designed for developing, training, and testing of AI models under in simulated real-world industrial scenarios. The framework enables iterative model refinement, accurate testing, and seamless integration with Industrial Internet-of-Things (IIoT) systems by integrating Apache Zeppelin [87], Eclipse Arrowhead [88], and IIoT data streams tools. Its features include dynamic interaction with live data, interoperability with existing IIoT frameworks, reduced deployment risks by model testing in production-like conditions, enhanced model robustness through real-time performance evaluations, and accelerated software development.

The AI-gym architecture as depicted in Figure 10 consists of three-layers that combines Apache Zeppelin, Eclipse Arrowhead, and IIoT components:

The Model Development Layer focuses on AI model development and quality assurance. Apache Zeppelin – the open-source notebook developed for collaborative data analysis – enables developers to write, test, and visualize their code

interactively. Python, Scala, and various other supported interpreters allow experimentation with a variety of machine learning frameworks. Developers can iteratively refine their models with feedback from data visualizations and performance metrics. Zeppelin's integrated quality checks help catching bugs before proceeding to the next stage of the AI development pipeline.

The Model Integration Layer uses the Eclipse Arrowhead Adapter to incorporate the AI models into larger IIoT ecosystems. The role of this layer is to connect Zeppelin to Eclipse Arrowhead's service-oriented architecture. The adapter merges AI solutions into Arrowhead's local cloud IIoT ecosystem, efficiently orchestrating data flow and service discovery. This layer also handles the security and interoperability functions, ensuring that the AI models adhere to industry standards while interacting with IIoT devices and frameworks.

Finally, the IIoT Data Layer tests the AI models under realistic conditions using Eclipse Arrowhead Local Cloud's IIoT data streams. This layer simulates production-like environments where AI models are exposed to dynamic inputs from IIoT devices: sensors, actuators, and industrial machines. The models' performance, scalability, and reliability are tested under varying conditions. Only authorized services can interact with the models, safeguarding sensitive data and operations by using Arrowhead's secure and modular architecture.

V. CONCLUSIONS

This paper presented a qualitative multi-case view of how LLMs and AI copilots can be integrated into industrial engineering workflows within the AIMS5.0 project. The cases cover different stages of the industrial lifecycle and range from conceptual designs to proofs of concept and pilot implementations. Taken together, they show how LLM-based systems can support engineers through knowledge access, natural-language interaction, software assistance, and controlled interaction with industrial systems.

Several technical patterns recur across the cases. Retrieval-Augmented Generation and semantic search provide access to domain-specific knowledge, while multi-agent approaches extend LLMs from question answering toward service orchestration. Other cases use LLMs to generate engineering artifacts or simplify interaction with semantic technologies. As these systems move closer to executable code or operational services, verification, access control, simulation, and human supervision become increasingly important.

The implementation experience also reveals a common set of challenges. Industrial knowledge is often distributed across heterogeneous, evolving, and multimodal sources. This makes document preparation, indexing, version management, and data currency central concerns. Data confidentiality and fine-grained access control are equally important when proprietary information is involved. In addition, factual correctness remains difficult to guarantee in specialized domains. Integration with legacy systems can create further difficulties, especially when interfaces are poorly documented or inconsistent.

The maturity of the demonstrators differ across the shown use cases. Most of the solutions are still at the proof-of-concept stage, and some have reached pilot use. The findings

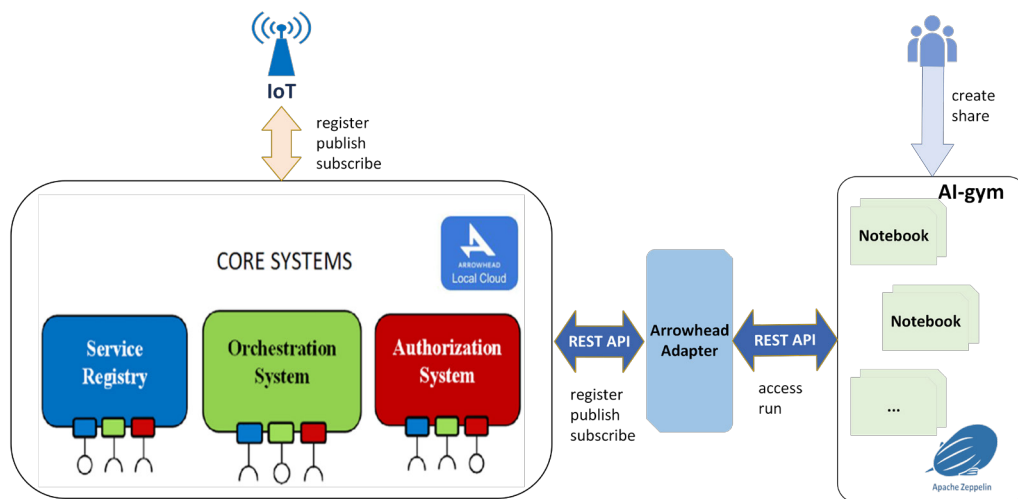


Fig. 10: AI-gym architecture.

of this paper are therefore best understood as implementation experience, design patterns, and practical lessons.

From an Industry 5.0 perspective, the clearest contribution of the presented systems is to human-centric engineering. They can make complex technical knowledge easier to access, reduce barriers to specialized tools, and support engineers while preserving human oversight. Some cases also contribute to resilience through verification, compliance support, controlled testing, and adaptation to heterogeneous systems. Sustainability effects are generally more indirect and arise through improved knowledge reuse, reduced rework, maintainability, and more efficient engineering processes. These effects require further empirical evaluation before broader conclusions can be drawn.

Future work should therefore focus on systematic evaluation in realistic industrial settings. Comparable measures are needed for response quality, task success, human effort, robustness, and operational impact. Further research is also required on secure and maintainable RAG architectures, domain adaptation, evolving knowledge bases, and verification of generated engineering artifacts. Longitudinal studies with industrial users would help distinguish projected benefits from effects that can be measured in routine operation.

Overall, the AIMS5.0 experience suggests that LLMs can become useful components of Industry 5.0 when they are embedded in well-designed technical and organizational safeguards. Their practical value depends not only on the capabilities of the language model, but also on the quality of the underlying knowledge, reliable system integration, explicit verification mechanisms, and clearly defined human responsibility. In this role, LLM-based systems are best viewed as tools that augment industrial expertise rather than replace it.

ACKNOWLEDGMENT

The research leading to these results was conducted in the AIMS5.0 project, which received funding from the Chips Joint Undertaking under grant agreement No. 101112089 and from the partners' national funding programmes.

REFERENCES

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: <http://arxiv.org/abs/1706.03762>
- [2] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.
- [3] R. Hendel, M. Geva, and A. Globerson, "In-context learning creates task vectors," *arXiv preprint arXiv:2310.15916*, 2023.
- [4] S. Lu, I. Bigoulaeva, R. Sachdeva, H. T. Madabushi, and I. Gurevych, "Are emergent abilities in large language models just in-context learning? arxiv," *arXiv preprint arXiv:2309.01809*, 2023.
- [5] C. Zhou, J. He, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Prompt consistency for zero-shot task generalization," *arXiv preprint arXiv:2205.00049*, 2022.
- [6] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja et al., "Multitask prompted training enables zero-shot task generalization," *arXiv preprint arXiv:2110.08207*, 2021.
- [7] Z. G. Yang and N. Ligeti-Nagy, "Improve performance of finetuning language models with prompting," *Infocommunications Journal*, vol. XV, no. Special Issue on Applied Informatics, pp. 62–68, 2023.
- [8] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.
- [9] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.
- [10] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou et al., "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.
- [11] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," 2023.
- [12] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," 2023.
- [13] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," *International Conference on Learning Representations (ICLR)*, Jan. 2023.
- [14] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "Pal: Program-aided language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 10 764–10 799.

- [15] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [16] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024. [Online]. Available: <https://arxiv.org/abs/2312.10997>
- [17] M. Antal, "Evaluation of embedding models for Hungarian question-answer retrieval on domain-specific and public benchmarks," *Infocommunications Journal*, vol. XVII, no. 4, pp. 2–11, 2025.
- [18] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," in *International Conference on Learning Representations (ICLR)*, 2024.
- [19] T. T. Procko and O. Ochoa, "Graph retrieval-augmented generation for large language models: A survey," in *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*, 2024, pp. 166–169.
- [20] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "Agentic retrieval-augmented generation: A survey on agentic rag," 2025. [Online]. Available: <https://arxiv.org/abs/2501.09136>
- [21] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, Sep. 2024, just Accepted. [Online]. Available: <https://doi.org/10.1145/3695988>
- [22] Z. Zhang, C. Chen, B. Liu, C. Liao, Z. Gong, H. Yu, J. Li, and R. Wang, "Unifying the Perspectives of NLP and Software Engineering: A Survey on Language Models for Code," p. arXiv:2311.07989, Nov. 2023.
- [23] X. Wang, B. Li, Y. Song, F. F. Xu, X. Tang, M. Zhuge, J. Pan, Y. Song, B. Li, J. Singh et al., "Openhands: An open platform for ai software developers as generalist agents," in *The Thirteenth International Conference on Learning Representations*, 2024.
- [24] T. Ridnik, D. Kreda, and I. Friedman, "Code generation with alphacodium: From prompt engineering to flow engineering," 2024. [Online]. Available: <https://arxiv.org/abs/2401.08500>
- [25] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirel, "The impact of ai on developer productivity: Evidence from github copilot," 2023. [Online]. Available: <https://arxiv.org/abs/2302.06590>
- [26] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," 2021. [Online]. Available: <https://arxiv.org/abs/2107.03374>
- [27] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "Fever: a large-scale dataset for fact extraction and verification," *arXiv preprint arXiv:1803.05355*, 2018.
- [28] A. Pagnoni, V. Balachandran, and Y. Tsvetkov, "Understanding factuality in abstractive summarization with frank: A benchmark for factuality metrics," *arXiv preprint arXiv:2104.13346*, 2021.
- [29] B. Goodrich, V. Rao, P. J. Liu, and M. Saleh, "Assessing the factual accuracy of generated text," in *proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 166–175.
- [30] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.
- [31] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi, "Factscore: Fine-grained atomic evaluation of factual precision in long form text generation," *arXiv preprint arXiv:2305.14251*, 2023.
- [32] A. Wang, K. Cho, and M. Lewis, "Asking and answering questions to evaluate the factual consistency of summaries," *arXiv preprint arXiv:2004.04228*, 2020.
- [33] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020.
- [34] W. Kryściński, B. McCann, C. Xiong, and R. Socher, "Evaluating the factual consistency of abstractive text summarization," *arXiv preprint arXiv:1910.12840*, 2019.
- [35] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano et al., "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.
- [36] S. Han, H. Schoelkopf, Y. Zhao, Z. Qi, M. Riddell, L. Benson, L. Sun, E. Zubova, Y. Qiao, M. Burtell et al., "Folio: Natural language reasoning with first-order logic," *arXiv preprint arXiv:2209.00840*, 2022.
- [37] J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang, "Logiqa: A challenge dataset for machine reading comprehension with logical reasoning," *arXiv preprint arXiv:2007.08124*, 2020.
- [38] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.
- [39] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: <https://aclanthology.org/W04-1013>
- [40] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez et al., "Chatbot arena: An open platform for evaluating llms by human preference, 2024," URL <https://arxiv.org/abs/2403.04132>, vol. 2, no. 9, 2024.
- [41] Y. Huang, Q. Zhang, L. Sun et al., "Trustgpt: A benchmark for trustworthy and responsible large language models," *arXiv preprint arXiv:2306.11507*, 2023.
- [42] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang et al., "Agentbench: Evaluating llms as agents," *arXiv preprint arXiv:2308.03688*, 2023.
- [43] S. Dahdal, L. Colombi, M. Brina, A. Gilli, M. Tortonesi, M. Vignoli, and C. Stefanelli, "An MLOps framework for GAN-based fault detection in Bonfiglioli's EVO plant," *Infocommunications Journal*, vol. XVI, no. 2, pp. 2–10, 2024.
- [44] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, and X. Hu, "Harnessing the power of llms in practice: A survey on chatgpt and beyond," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 6, pp. 1–32, 2024.
- [45] A. Mailach, S. Simon, J. Dorn, and N. Siegmund, "Practitioners' discussions on building llm-based applications for production," *arXiv preprint arXiv:2411.08574*, 2024.
- [46] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu et al., "Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *arXiv preprint arXiv:2405.10825*, 2024.
- [47] Y. Xia, J. Kim, Y. Chen, H. Ye, S. Kundu, C. Hao, and N. Talati, "Understanding the performance and estimating the cost of llm finetuning," *arXiv preprint arXiv:2408.04693*, 2024.
- [48] S. Shekhar, T. Dubey, K. Mukherjee, A. Saxena, A. Tyagi, and N. Kotla, "Towards optimizing the costs of llm usage," *arXiv preprint arXiv:2402.01742*, 2024.
- [49] Y. Ma, S. Zheng, Z. Yang, P. Zheng, J. Leng, and J. Hong, "Leveraging large language models in next generation intelligent manufacturing: Retrospect and prospect," *Journal of Manufacturing Systems*, vol. 82, pp. 809–840, 2025.
- [50] C. Zhang, Q. Xu, Y. Yu, G. Zhou, K. Zeng, F. Chang, and K. Ding, "A survey on potentials, pathways and challenges of large language models in new-generation intelligent manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 92, p. 102 883, 2025.
- [51] Y. Li, H. Zhao, H. Jiang, Y. Pan, Z. Liu, Z. Wu, P. Shu, J. Tian, T. Yang, S. Xu, Y. Lyu, P. Blenk, J. Pence, J. Rupram, E. Banu, N. Liu, L. Wang, W. Song, X. Zhai, K. Song, D. Zhu, B. Li, X. Wang, and T. Liu, "Large language models for manufacturing," *arXiv preprint arXiv:2410.21418*, 2024.
- [52] M. Sadeqi Bajestani, D. Mun, and D. Kim, "Human-in-the-loop and large language models in smart manufacturing: Current applications, challenges, and perspectives," *Journal of Manufacturing Systems*, vol. 86, pp. 913–941, 2026.
- [53] H. Kozirolek, S. Gruener, and V. Ashiwal, "ChatGPT for PLC/DCS control logic generation," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8, arXiv:2305.15809.
- [54] M. Fakhri, R. Dharmaji, Y. Moghaddas, G. Quiros, T. Ogundare, and M. A. Al Faruque, "LLM4PLC: Harnessing large language models for verifiable programming of PLCs in industrial control systems," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2024, arXiv:2401.05443.

- [55] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500. [Online]. Available: <https://www.semanticscholar.org/paper/91deaf9d324c8feafc189da0da03e60a60287bca>
- [56] Y. Xia, M. Shenoy, N. Jazdi, and M. Weyrich, "Towards autonomous system: flexible modular production system enhanced with large language model agents," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8.
- [57] J. Lim, B. Vogel-Heuser, and I. Kovalenko, "Large language model-enabled multi-agent manufacturing systems," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024, pp. 3940–3946.
- [58] D. Patel, S. Lin, J. Rayfield, N. Zhou, C. Shyalika, S. R. Yarrabothula, R. Vaculin, N. Martinez, F. O'Donncha, and J. Kalagnanam, "AssetOps-Bench: A real-world evaluation benchmark for AI-driven task automation in industrial asset management," *arXiv preprint arXiv:2506.03828*, 2025.
- [59] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney *et al.*, "ChipNeMo: Domain-adapted LLMs for chip design," *arXiv preprint arXiv:2311.00176*, 2023.
- [60] E. Brynjolfsson, D. Li, and L. Raymond, "Generative AI at work," *The Quarterly Journal of Economics*, vol. 140, no. 2, pp. 889–942, 2025.
- [61] Siemens AG, "Siemens introduces AI agents for industrial automation," Press release, Automate 2025, Detroit, 12 May 2025. <https://www.automation.com/article/siemens-ai-agents-industrial-automation>, 2025.
- [62] B. Martini, D. Bellisario, and P. Coletti, "Human-centered and sustainable artificial intelligence in Industry 5.0: Challenges and perspectives," *Sustainability*, vol. 16, no. 13, p. 5448, 2024.
- [63] A. Frankó, G. Hollósi, D. Ficzer, and P. Varga, "Synchronized real-time communication in the Eclipse Arrowhead framework," *Infocommunications Journal*, vol. XVIII, no. 2, pp. 91–98, 2026.
- [64] "Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence (artificial intelligence act)," <https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng>, 2024, official Journal of the European Union, L 168, 12 July 2024.
- [65] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
- [66] DeepSeek-AI, "DeepSeek-V3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.
- [67] D. Guo *et al.*, "DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning," *Nature*, vol. 645, pp. 633–638, 2025.
- [68] A. Yang *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.
- [69] OpenAI, "gpt-oss-120b & gpt-oss-20b model card," *arXiv preprint arXiv:2508.10925*, 2025.
- [70] P. Smeros, V. Emonet, R. Wang, A.-C. Sima, and T. Mendes de Farias, "SPARQL-LLM: Real-time SPARQL query generation from natural language questions," *arXiv preprint arXiv:2512.14277*, 2025.
- [71] S. Tramp, M. Gölo, E. Marx, and P. V. do Carmo, Eds., *Proceedings of the First International TEXT2SPARQL Challenge 2025, co-located with Text2KG at ESWC 2025*, ser. CEUR Workshop Proceedings, vol. 4094. Portorož, Slovenia: CEUR-WS.org, 2025. [Online]. Available: <https://ceur-ws.org/Vol-4094/>
- [72] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: <https://arxiv.org/abs/1810.04805>
- [73] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia," in *The Semantic Web-ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II* 18. Springer, 2019, pp. 69–78.
- [74] A. Lonkhang and J. Botzheim, "Path planning transformer supervised by improved RRT* with reduced random map size for mobile robots," *Infocommunications Journal*, vol. XVIII, no. 1, pp. 100–109, 2026.
- [75] Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang, "The dawn of llms: Preliminary explorations with gpt-4v(ision)," 2023. [Online]. Available: <https://arxiv.org/abs/2309.17421>
- [76] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock *et al.*, "Flamingo: a visual language model for few-shot learning," in *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., 2022.
- [77] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," 2023. [Online]. Available: <https://arxiv.org/abs/2301.12597>
- [78] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of The 7th Conference on Robot Learning*, ser. PMLR, vol. 229, 2023, pp. 2165–2183.
- [79] M. J. Kim *et al.*, "Openvla: An open-source vision-language-action model," in *Proceedings of The 8th Conference on Robot Learning*, ser. PMLR, vol. 270, 2025, pp. 2679–2713.
- [80] A. Béres and B. Gyires-Tóth, "Enhancing visual domain randomization with real images for sim-to-real transfer," *Infocommunications Journal*, vol. XV, no. 1, pp. 15–25, 2023.
- [81] T. Berners-Lee, "The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, 2001. [Online]. Available: <https://api.semanticscholar.org/CorpusID:265709654>
- [82] H. Ehm, E. Schoitsch, J. W. van der Weit, N. Ramzy, L. Luo, and D. L. Gruetzner, "Digital reference: a quasi-standard for digitalization in the domain of semiconductor supply chains," *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1, pp. 563–570, 2020. [Online]. Available: <https://api.semanticscholar.org/CorpusID:228092656>
- [83] H. B. Giglou, J. D'Souza, and S. Auer, "Llms4ol: Large language models for ontology learning," *ArXiv*, vol. abs/2307.16648, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:260334867>
- [84] H. B. Giglou, J. D'Souza, F. Engel, and S. Auer, "Llms4om: Matching ontologies with large language models," *ArXiv*, vol. abs/2404.10317, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:269157258>
- [85] G. Naidu, T. Zuva, and E. M. Sibanda, "A review of evaluation metrics in machine learning algorithms," in *Artificial Intelligence Application in Networks and Systems*, ser. Lecture Notes in Networks and Systems, R. Silhavy and P. Silhavy, Eds. Cham: Springer International Publishing, 2023, vol. 724, pp. 15–25.
- [86] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024.
- [87] T. A. S. Foundation, "Zeppelin," 23/10/2024. [Online]. Available: <https://zeppelin.apache.org/>
- [88] "Eclipse arrowhead – eclipse arrowhead framework and implementation platform," 25/11/2024. [Online]. Available: <https://arrowhead.eu/>