AI-Powered Architecting for Industry 4.0 Cyber-Physical Production Systems: A Novel Approach, Research Problems and Challenges

Stephen John Warnett[1,2][0000−0003−0650−0981] and
Uwe Zdun[1][0000−0002−6233−2591]

[1] Research Group Software Architecture, Faculty of Computer Science, University of Vienna, Vienna, Austria
[2] UniVie Doctoral School Computer Science DoCS, Faculty of Computer Science, University of Vienna, Vienna, Austria
{stephen.warnett|uwe.zdun}@univie.ac.at

Abstract. The integration of artificial intelligence (AI) into software architecture has the potential to greatly support practitioners in architecting machine learning (ML) and reinforcement learning (RL) systems. In the context of Industry 4.0 cyber-physical production systems (CPPSs), this work presents our approach, research problems, and open challenges when designing ML and RL systems and applying AI-powered architecting to enhance automation and reduce manual effort. We describe a novel approach with six advanced contributions: organising architectural design decisions (ADDs) linked to the ML workflow and deployment, extending machine learning operations (MLOps) into reinforcement learning operations (RLOps), introducing formal architectural model development, adding automatic conformance checks, using large language models (LLMs) in RL conformance assessment and creating MLOps and RLOps pipelines using LLMs and a low-code approach. We also describe how earlier contributions in supporting the architecting process could be enhanced using AI. We discuss four further key open research challenges: RL model versioning, the development of RL communication protocols, working with agentic AI, and supporting RL within continuous integration and continuous delivery pipelines.

Keywords: artificial intelligence · MLOps · RLOps · machine learning · reinforcement learning · software architecture · software engineering · cyber-physical production systems

1 Introduction

Integrating artificial intelligence (AI) into software architecture can catalyse rapid advancements in how practitioners design, optimise, and maintain systems [1]. AI-driven techniques transform traditional practices by documenting decision-making processes, refining architectural trade-offs, and enabling dynamically adaptive systems capable of self-evolution [2,3].

AI's role in overcoming long-standing issues in software architecture, including complexity management, up-to-date documentation, and continuous evolution of systems, is becoming increasingly significant [1]. The manual expertise, effort, and sophisticated reasoning required when applying traditional architecting practices cannot keep up with the demanding architectural needs and operational uncertainty of AI-enabled systems [1]. Combined with the highly dynamic, rapidly changing, and unpredictable nature of the domains in which architectures are applied, for instance in cyber-physical production systems (CPPSs) [4], there is an urgent need to investigate how AI can automate the architectural decision-making process that caters for such domains [1,5,6] in machine learning (ML) [7] and reinforcement learning (RL) [8]-based systems.

In this paper, we examine architecting ML and RL-based systems in the context of our recent research. We identify open research challenges encountered when architecting such systems and consider how AI may be applied to the architectural process. We also report on how we have begun to utilise large language models (LLMs) in addressing research problems to aid practitioners when architecting ML and RL-based systems, present some forthcoming work, and identify open research challenges we plan to address in future work.

We describe our overarching approach, covering topics including the cataloguing of architectural design decisions (ADDs) for the ML workflow and ML deployment, as well as RL training strategies in an Industry 4.0 [9] setting, together with the analysis and assessment of quality aspects in machine learning operations (MLOps) and reinforcement learning operations (RLOps) architectures. We discuss how associated research problems and manual techniques developed to support ML and RL practitioners could be automated by leveraging AI and how AI could replace these manual methods by generating MLOps architectures and providing continuous and self-adapting architectural guidance based on evolving needs. We also describe the problem of assessing projects for RL ADDs and our first foray into LLMs. We describe forthcoming work that utilises LLMs with a low-code approach and a human-in-the-loop to integrate AI into MLOps and RLOps pipeline architecture generation and evolution for open-source projects and an Industry 4.0 case study. Finally, we discuss open challenges and new research opportunities where AI can be applied to software architecting that we will address in the future in collaboration with our Industry 4.0 project partners.

The rest of the paper is structured as follows. In Section 2, we introduce key concepts and terminology integral to the topics discussed in the paper. Section 3 describes our approach in the context of research problems and our contributions. In Section 4, we introduce open research challenges that we plan to address in the context of our Industry 4.0 project collaboration, and in Section 5, we discuss specific aspects of our approach and contributions. We conclude the paper in Section 6.

2 Key Concepts and Terminology

This section briefly describes key relevant concepts and terminology, and how they are correlated.

Architectural design decisions (ADDs) are concerned with the series of decisions upon which software architecture is based. Making architectural decisions has a long-term impact on the system and covers decisions that are costly to change. Architects must prioritise, since making an architectural decision involves trade-offs concerning requirements. ADDs are essential for software architecting, since they capture the architectural choices made during the early design process [10,11].

MLOps [12,13] represents a foundational paradigm in contemporary AI engineering that has evolved to address the complexities of deploying and maintaining ML systems in production environments. It constitutes a holistic approach to the ML workflow, applies DevOps [14] and continuous integration/continuous delivery (CI/CD) [15] principles to ML, and involves training, deploying, and managing supervised and unsupervised ML models in production environments [16,17].

RLOps [18] has emerged as a distinct but related paradigm that still represents a critical gap in current academic literature and industrial practice [5]. The extent to which MLOps practices apply to RL, particularly due to major differences between ML and RL concerning model deployment and model training, is still not fully understood. We addressed this deficiency and demystified RLOps by investigating how MLOps principles can be extended to RL contexts through an empirical Industry 4.0 case study focusing on CPPSs [5] with our Industry 4.0 project partners.

Large language models (LLMs)[3] are trained using methods that do not require explicit labelling of the training data. Self-attention in transformer structures helps models deal with long sequences of data by processing it in bulk so that models can use a large number of parameters and access large datasets [19][4].

Industry 4.0 and cyber-physical production systems (CPPSs) [9,20] represent a combination of digital technologies symbolising a significant development in modern manufacturing. Smart factories in Industry 4.0 combine the Internet of Things, AI, and Cloud Computing and are efficient networks interacting automatically and sharing data in real time.

A systematic alignment of technical decisions and operational strategies can be achieved by utilising ADDs as a basis for the architectural generation process for MLOps [21]. Involving LLMs in the architectural generation process increases its efficiency by automating various aspects of architectural synthesis, documentation, and adaptation to changing needs [22]. Such strengths are particularly impactful when transferred to RLOps since RL requires a flexible and resilient architecture [18]. Implemented in the context of Industry 4.0 and CPPSs, these strengths can be used to architect intelligent and interconnected manufacturing

[3] https://en.wikipedia.org/wiki/Large_language_model
[4] https://www.nvidia.com/en-us/glossary/large-language-models

environments quickly and continuously improve the development of operational excellence and innovation in complex digital-physical ecosystems [23].

3 Approach

We followed a systematic approach to understand and support how practitioners can better design, build, and manage ML/RL systems in an automated fashion, using AI where appropriate. Our approach has evolved, starting with asking fundamental questions about practitioners' understanding of architectural decisions, and developing into implementing tools in an Industry 4.0 context for CPPSs that utilise AI and can help practitioners make better decisions and automate specific tasks.

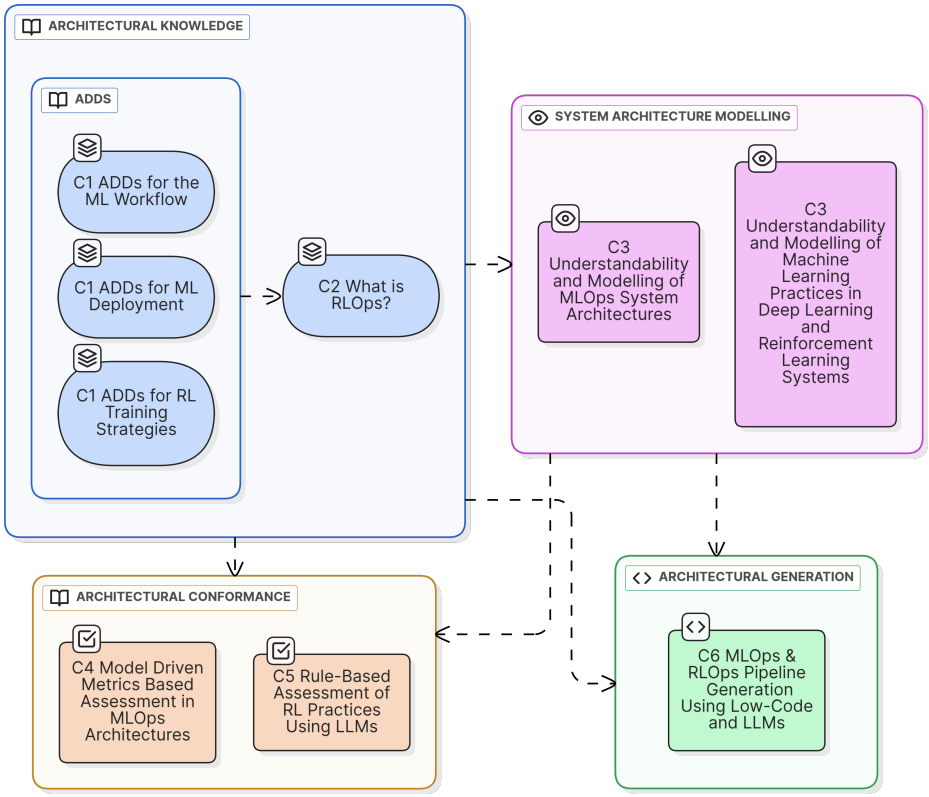

Fig. 1. Overview of Our Approach.

Figure 1 depicts our overarching approach, which consists of several contributions. Fundamental to the overall strategy was the initial consolidation of

Architectural Knowledge in the form of ADDs for ML and RL and an understanding of emerging paradigms, such as RLOps. We then used this knowledge to develop a **System Architecture Modelling** method, and tools for automating the assessment of **Architectural Conformance** using detectors and LLMs. **Architectural Knowledge** is also applied to an **Architectural Generation** technique that we developed, using LLMs in a low-code approach. Below, we discuss the various research problems we have addressed so far and outline the details of our contributions.

ADDs for ML and RL

Problem P1 Productionising ML models is technically challenging. This challenge is compounded by knowledge gaps on the level of the individual practitioner and the lack of formalised, reusable architectural knowledge sources. For RL, practitioners have many training, deployment, and coordination strategies to choose from.

Contribution C1 We systematically studied architectural knowledge by conducting three Straussian grounded theory [24] studies of practitioner (grey) literature in various subfields of AI, including the ML workflow, ML deployment, and RL training strategies [16,17,25]. The aim was to catalogue ADDs and current practices and provide design guidance for practitioners since existing practitioner guidelines and best practices are often informal and inconsistent. We developed a novel ADD model and established reusable architectural knowledge in the form of ADDs, which form the foundation of our overarching approach and help bridge the gap between science and practice. Our formal ADD Python models are visualised in automatically generated UML-based model diagrams.

What is RLOps?

Problem P2 The extent to which MLOps and architectural practices apply to RL is poorly understood.

Contribution C2 We conducted an exploratory, qualitative, deductive-inductive industry case study [5] on an Industry 4.0 CPPS and performed content analysis [26] of CPPS artefacts like architectural schematics and source code to understand their relation to known ADDs and associated decision options. In doing so, we fostered an initial understanding of RLOps architectures and provided a reference for practitioners that can be used as design guidance.

MLOps, ML and RL System Architecture Modelling and Understandability

Problem P3 How to model ML and RL-based systems so that they are understandable, can be reasoned about, and programmatically analysed?

Contribution C3 We developed a modelling method for ML and RL-based systems by defining a formal metamodel with appropriate stereotype extensions in Python and automatically generating UML-based visualisations of the modelled systems. We systematically developed the metamodel and modelled freely available system architectures, open source projects, and relevant aspects of our

Industry 4.0 partner's CPPS. We conducted user studies in the form of controlled experiments [27,28] to validate how well UML diagrams generated from our models aid system architecture and design comprehension.

Assessing Support for Quality Aspects in MLOps System Architectures
Problem P4 Ensuring conformance to ADDs, patterns, practices, and decision options in MLOps systems is a laborious and error-prone manual task. Objective measures, like metrics, for assessing the ADD options are also lacking.
Contribution C4 To avoid manual validation of architectural conformance, we implemented detectors and novel, technology-agnostic metrics to automatically calculate conformance of modelled MLOps system architectures to known ADDs in the context of quality criteria, such as the level of support for automation practices, in given architectures [29]. We statistically validated our solution, showing that it can be used to assess MLOps architectures programmatically.

Assessing RL Practices Using LLMs
Problem P5 As RL-based systems become increasingly complex, there is a need for standardised and automated detection of conformance to best practices, notably concerning training methods.
Contribution C5 We implemented a rule-based framework that integrates LLMs and heuristic-based code detectors to ensure compliance with best practices in RL training pipelines [30]. Our architectural rules focus on best practices in RL-based architectures that are particularly relevant to CPPSs, like checkpoints, hyperparameter tuning, and agent configuration. We validated our solution via open-source projects and an Industry 4.0 CPPS case study.

Generating MLOps and RLOps Pipelines Architectures using Low-Code and LLMs
Problem P6: Automating the ML/RL workflow requires specialist expertise and considerable manual effort to write and maintain pipeline configurations in an error-prone process that hinders scalability and rapid deployment.
Contribution C6 In forthcoming work, which has been completed but not yet published, we present a new solution for generating pipelines for MLOps and RLOps. Our solution is a low-code, template-based approach that leverages LLMs and a human-in-the-loop to automate pipeline generation, validation, and deployment. We evaluated our solution against seven LLMs, four usage scenarios, three open source projects, and an Industry 4.0 CPPS case study, achieving very low error rates across several metrics. Our solution has the potential to enable rapid scaling and deployment of reliable RLOps pipelines and negates the need for practitioners to have advanced software engineering or DevOps skills.

3.1 Placement of Our Approach in a CI/CD Pipeline

Figure 2 shows how our overall approach fits into the context of CI/CD pipelines for ML and RL. The figure depicts high-level essential steps in a CI/CD pipeline

common to ML and RL, which are coloured blue, orange, green, and purple, augmented by contributions from our approach, which are coloured yellow.

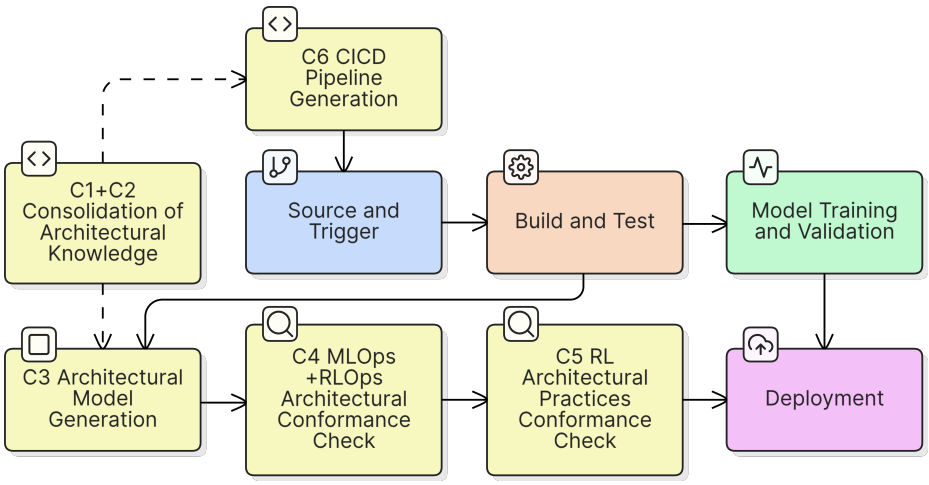

Fig. 2. Placement of Our Approach in a CI/CD Pipeline.

The basis for the entire process **C1+C2 Consolidation of Architectural Knowledge**. It is an activity that only needs to be performed rarely, as new practices and patterns are not likely to be documented in grey literature very often. Initially, the pipeline is triggered in the usual way, such as when changes to the source code are made or an external component triggers the pipeline, for example, on model performance drift (**Source and Trigger**). In an Industry 4.0 CPPS setting, where frequent changes are made to pipeline configurations, and new pipelines need to be rapidly generated at scale, we can trigger a new pipeline with **C6 CICD Pipeline Generation** when our LLM-based low-code tool generates a new pipeline configuration and commits it, for instance, to a GitLab repository.

Following the **Build and Test** phase, both **Model Training and Validation** and **C3 Architectural Model Generation** can run in parallel. The latter phase uses **C1+C2 Consolidation of Architectural Knowledge** as a basis. Currently, **C3 Architectural Model Generation** is a manual activity due to the distributed and polyglot nature of ML/RL-enabled systems, which makes it hard to create formal models from source code or architectural schematics using traditional methods such as source code parsing and requires considerable expertise in a time-consuming process. However, with modern AI tools, we think system architecture model generation could be automated, and we discuss this further in Section 5.

The next step is **C4 MLOps+RLOps Architectural Conformance Check**, which is an automated task that checks for architectural conformance, based on the provided system model, to specific quality attributes under consideration of specific ADDs and associated practices and patterns. This task is manually programmed, but we envisage an enhancement using similar techniques to the next step, **C5 RL Architectural Practices Conformance Check**, where we use LLMs combined with the consolidated **Architectural Knowledge**. We describe this proposed improved process in Section 5. **C5 RL Architectural Practices Conformance Check** runs on RL source code and checks for the presence of specific architectural practices. Assuming the conformance checks, model training, and validation succeeded, we reach the **Deployment** stage, and the pipeline can deploy the model.

4 Open Research Challenges

In collaboration with our Industry 4.0 partners, we have identified several open research challenges that must be addressed. Due to the combination of ML, RL, and Industry 4.0 technologies, software architecture obstacles demand new solutions. This section examines four significant challenges for building reliable and scalable architectures for intelligent manufacturing systems and how AI may help.

Model Versioning for RL in CPPSs
Challenge Traditional methods for managing model versions are not suited to RL in smart factories due to continuous learning by the agents, and this calls for more advanced tools to oversee model updates, track policy changes, and update models to fit the smart factory environment whilst maintaining operational stability. Difficulties arise when several agents are distributed across different parts of a production line and require simultaneous, jointly managed version updates and rollbacks.
Strategy In prior work, we noted the use of a hybrid MLOps-RLOps solution that updates model registries with RL policy metadata, exploration strategies, and environment interaction history [5]. The solutions involve using distributed version control to track model parameters and learned policies, and the setup of rollback mechanisms [31]. AI-powered automated version management might choose the most effective model using AI meta-learning techniques and analysing prior performance records and the current context.

Communication Protocols for RL in Industry 4.0
Challenge Current communication schemas, such as the Model Context Protocol (MCP)[5], whilst useful for AI-tool interactions, do not cover necessary architectural designs used in industrial fields where critical operations, real-time constraints, and legacy systems predominate. Protocol implementations mostly handle desktop applications and simple AI assistants, and they do not cover the

[5] https://www.anthropic.com/news/model-context-protocol

needs of industry, which require reliability, rigid response times, and compatibility with current manufacturing systems.
Strategy One strategy is to work on system architectures built for industry, adding real-time guarantees, secure authentication and manufacturing-oriented MCP-like protocol architectures[6]. A solution could build federated networks that cross multiple security domains in a factory, use special tools for data transfer over industrial networks and facilitate interoperability of networks with other popular industrial protocols. AI could also help facilitate protocol optimisation as network and production changes occur.

Agentic AI Architecture for CPPSs
Challenge Using agentic AI in CPPSs means building architectures that can handle complex decisions on their own, whilst making sure the AI is secure, reliable and works with previous control systems [32][7]. Currently, agentic systems mainly concentrate on software environments and fail to include industry-relevant error handling, latency assurances and reliability guarantees[8]. It is essential to ensure that changing behaviours of autonomous agents do not disrupt the orderly operations of manufacturing [33].
Strategy A strategy could emphasise architectures that ensure safety-related functions are under separate control from autonomous agents, using advanced verification techniques on agent systems and developing hybrid human-agent supervision models [32]. Some ways to reduce risks are to precisely define agent human responsibilities, standardise interfaces between agent systems and standard industrial controls, and to include continuous monitoring and intervention methods [31]. AI-assisted optimisation in architecture could influence agent autonomy according to risks and outcomes in production.

Support for RL in CI/CD Pipelines
Challenge Existing frameworks for CI/CD fail to adequately support RL systems since they do not support many of the unique needs of RL, like those described in the challenges above (versioning for RL models, support for RL communication protocols, and agentic AI systems). The need is for CI/CD pipelines that understand complex deployment environments and can learn system evolution [31] since traditional software deployment methods do not consider the unique demands of learning and adaptation after release [5].
Strategy One strategy is to build multi-step deployment systems for RL artefacts, create automated systems to test agent skills, and design monitoring tools that notice changes in production environments [31]. AI-supported pipeline configuration can bootstrap the appropriate deployment methods given the model type, risks, and production needs, with feedback loops that improve the system in controlled production settings [5].

[6] https://www.linkedin.com/pulse/securing-model-context-protocol-mcp-architecture-best-srivastava-v4hgf
[7] https://techcommunity.microsoft.com/blog/machinelearningblog/baseline-agentic-ai-systems-architecture/4207137
[8] https://www.ibm.com/think/topics/agentic-architecture

5 Discussion

In Section 3, we described our overall approach. We noted that some of the solutions we developed to address specific challenges earlier in our research could potentially be enhanced due to recent progress that has been made in AI. This section discusses how AI could improve some residual manual aspects of our approach. Concrete details, such as pipeline examples, architectures, controlled experiment results, limitations of our approach, and threats to validity for each contribution, are discussed in the respective papers. We also reference replication packages for our prior contributions in the corresponding papers.

ML models can now scan technical documentation, find architectural patterns and decisions made in the design, and group system elements based on source code and architectural diagrams[9][10]. Recent studies show that LLMs and generative AI can significantly improve grounded theory analysis by automating coding, finding patterns, and extracting themes without compromising the accuracy of qualitative work. For example, Übellacker [34] describes AcademiaOS, which is an initial attempt to automate grounded theory using LLMs by applying the understanding, generation, and reasoning capabilities of LLMs to augment humans in qualitative research. Yue et al. [35] apply ChatGPT-4 Turbo to grounded theory and describe how LLMs can enhance the efficiency of text coding and qualitative analysis in grounded theory. When gathering **Architectural Knowledge** (**C1** and **C2**), rather than periodically manually performing grounded theory and content analysis studies, which can be time-consuming, we could consider reducing manual work by using AI tools to implement alternative solutions. An advantage is that this knowledge-gathering step could be more easily automated within a CI/CD pipeline. The gathered knowledge is more frequently updated to use in later CI/CD steps that implement our approach and make use of this knowledge, such as **C3 Architectural Model Generation** and the **C4 MLOps+RLOps Architectural Conformance Check**.

LLMs can now build structural and behavioural models for systems without relying on human-written code. Latest findings also suggest that LLMs now allow systems to be modelled, for example, with SysML[11] diagrams automatically, helping lower the need to do this work manually. Apvrille and Sultan [36] describe how LLMs can make sense of natural text descriptions and generate content fitting predefined formats, supporting the automation of model generation. They introduce a framework where LLMs automatically construct structural and behavioural SysML diagrams from system specifications. **Contribution C3 Architectural Model Generation** in Figure 2 is currently a manual process in which the practitioner models the system architecture in Python using our extensible MLOps/RLOps metamodel. This formal system model can then be processed in later stages of the pipeline, for instance during the **C4 MLOps+RLOps Architectural Conformance Check**. Rather than gener-

[9] https://www.westat.com/machine-learning-nlp
[10] https://mindthegraph.com/blog/automated-content-analysis
[11] https://sysml.org

ating diagrams directly, we envisage an enhanced solution that would make use of generative AI to write Python code, based on our CodeableModels[12] metamodel, to generate the formal system architecture models. This automated stage would dramatically reduce the required modelling effort in the early stages of a project. Optionally, the tool could subsequently generate UML visualisations from the Python models using PlantUML[13] as it currently does.

Contribution C4 (MLOps+RLOps Architectural Conformance Check) substantially reduces the need for manual assessment of MLOps/RLOps architectures for conformance to known ADDs and decision options and allows for automation in a CI/CD pipeline. However, some manual effort is still involved in implementing the detectors that analyse the models, and the metrics that evaluate the support for quality aspects based on the applied decision options. An upgraded solution could use LLMs similarly to our implementation of **C5 RL Architectural Practices Conformance Check** by examining the Python models generated using **C3 Architectural Model Generation** based on the Architectural Knowledge from **C1+C2 Consolidation of Architectural Knowledge**. Recent related work describes applying LLMs to architectural conformance checking. Keshri et al. [37] present a novel system for verifying code implementations against the algorithms and methodologies from corresponding research papers. Their solution uses retrieval-augmented generation [38] to extract relevant details from both the research papers and code bases, and compares them using LLMs to reduce manual effort, enhance research credibility, and advance the state of the art in code verification.

6 Conclusion

This paper outlines our approach towards using AI for architecting ML and RL-based systems, particularly for Industry 4.0 CPPSs. We outlined six research problems and how we addressed them through Straussian grounded theory-based grey literature studies, the development of a novel ADD metamodel in Python, an exploratory industry case study on an Industry 4.0 CPPS, the development of a novel, formal metamodel for modelling ML and RL-based systems, controlled experiments testing whether providing semi-formal system and architecture representations as UML diagrams helps understanding of MLOps system architectures compared to informal system architecture diagrams, a method for automatically checking conformance of MLOps architectures to known ADDs, the assessment of conformance to RL best practices in RL code using LLMs, and the generation of MLOps and RLOps pipeline architecture using low-code and LLMs. We showed where we have already applied AI techniques, and described how we could further enhance our approach with AI.

Using our approach in CI/CD pipelines helps automate much of the process. Still, some significant open challenges remain: advanced ways to version models for distributed RL agents, effective communication protocols for industrial

[12] https://github.com/uzdun/CodeableModels
[13] https://github.com/plantuml/plantuml

applications, safe integration of agents in AI systems, and RL integration into CI/CD pipelines. We described these four open challenges and possible strategies to address them. Future work will address the identified open challenges and further automate our approach in collaboration with industry partners. Our research helps to form a basis for intelligent, adaptive systems that can constantly adapt to fit the dynamic demands of modern production systems and lead to functional AI implementation in industry when architecting ML and RL-based systems.

Acknowledgements. This work was supported by the FFG (Austrian Research Promotion Agency) project MODIS (no. FO999895431). This research was funded in whole or in part by the Austrian Science Fund (FWF) project CQ4CD, Grant-DOI: 10.55776/I6510. For open access purposes, the authors have applied a CC BY public copyright licence to any author-accepted manuscript version arising from this submission.

References

1. Bucaioni, A., Weyssow, M., He, J., Lyu, Y., Lo, D.: Artificial Intelligence for Software Architecture: Literature Review and the Road Ahead. In: 2030 Software Engineering - 2025 (June 2025), http://www.ipr.mdu.se/publications/7175-
2. Zhou, X., Li, R., Liang, P., Zhang, B., Shahin, M., Li, Z., Yang, C.: Using LLMs in Generating Design Rationale for Software Architecture Decisions (2025), https://arxiv.org/abs/2504.20781
3. Gheibi, O., Weyns, D., Quin, F.: Applying Machine Learning in Self-adaptive Systems: A Systematic Literature Review. ACM Trans. Auton. Adapt. Syst. **15**(3) (Aug 2021). https://doi.org/10.1145/3469440
4. Monostori, L.: Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. Procedia CIRP **17**, 9–13 (2014). https://doi.org/10.1016/j.procir.2014.03.115, https://www.sciencedirect.com/science/article/pii/S2212827114003497, Variety Management in Manufacturing
5. Warnett, S.J., Zdun, U.: Bridging the Gap Between MLOps and RLOps: An Industry 4.0 Case Study on Architectural Design Decisions in Practice. In: 2025 IEEE 22nd International Conference on Software Architecture (ICSA). pp. 232–242 (2025). https://doi.org/10.1109/ICSA65012.2025.00031
6. Ramic, A., Kugele, S.: A Systematic Mapping Study on Software Architecture for AI-based Mobility Systems (2025), https://arxiv.org/abs/2506.01595
7. Jordan, M.I., Mitchell, T.M.: Machine learning: Trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015)
8. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
9. Singh, H., Singh, B.: Industry 4.0 technologies integration with lean production tools: a review. The TQM Journal (02 2024). https://doi.org/10.1108/TQM-02-2022-0065
10. Jansen, A., Bosch, J.: Software Architecture as a Set of Architectural Design Decisions. In: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05). pp. 109–120 (2005). https://doi.org/10.1109/WICSA.2005.61

11. Shahin, M., Liang, P., Li, Z.: Do architectural design decisions improve the understanding of software architecture? two controlled experiments. In: Proceedings of the 22nd International Conference on Program Comprehension. p. 3–13. ICPC 2014, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2597008.2597139
12. Kreuzberger, D., Kühl, N., Hirschl, S.: Machine Learning Operations (MLOps): Overview, Definition, and Architecture. ArXiv **abs/2205.02302** (2022)
13. Hewage, N., Meedeniya, D.: Machine Learning Operations: A Survey on MLOps Tool Support (2022). https://doi.org/10.48550/ARXIV.2202.10169
14. Bass, L., Weber, I., Zhu, L.: DevOps: A Software Architect's Perspective. Addison-Wesley Professional, 1st edn. (2015)
15. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional, 1st edn. (2010)
16. Warnett, S.J., Zdun, U.: Architectural Design Decisions for the Machine Learning Workflow. Computer **55**(3), 40–51 (2022). https://doi.org/10.1109/MC.2021.3134800
17. Warnett, S.J., Zdun, U.: Architectural Design Decisions for Machine Learning Deployment. In: 2022 IEEE 19th International Conference on Software Architecture (ICSA). pp. 90–100 (2022). https://doi.org/10.1109/ICSA53651.2022.00017
18. Li, P., Thomas, J., Wang, X., Khalil, A., Ahmad, A., Inacio, R., Kapoor, S., Parekh, A., Doufexi, A., Shojaeifard, A., et al.: RLOps: Development life-cycle of reinforcement learning aided open RAN. IEEE Access **10**, 113808–113826 (2022). https://doi.org/10.1109/ACCESS.2022.3217511
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
20. Khdoudi, A., Masrour, T., El Hassani, I., El Mazgualdi, C.: A Deep-Reinforcement-Learning-Based Digital Twin for Manufacturing Process Optimization. Systems **12**(2) (2024). https://doi.org/10.3390/systems12020038, https://www.mdpi.com/2079-8954/12/2/38
21. Leest, J., Gerostathopoulos, I., Raibulet, C.: Evolvability of Machine Learning-based Systems: An Architectural Design Decision Framework. In: 2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C). pp. 106–110 (2023). https://doi.org/10.1109/ICSA-C57050.2023.00033
22. Dhar, R., Vaidhyanathan, K., Varma, V.: Can LLMs Generate Architectural Design Decisions? - An Exploratory Empirical Study. In: 2024 IEEE 21st International Conference on Software Architecture (ICSA). pp. 79–89 (2024). https://doi.org/10.1109/ICSA59870.2024.00016
23. Fischbach, A., Strohschein, J., Bunte, A., Stork, J., Faeskorn-Woyke, H., Moriz, N., Bartz-Beielstein, T.: CAAI—a cognitive architecture to introduce artificial intelligence in cyber-physical production systems. The International Journal of Advanced Manufacturing Technology **111**, 1–18 (11 2020). https://doi.org/10.1007/s00170-020-06094-z
24. Corbin, J., Strauss, A.L.: Grounded theory research: Procedures, canons, and evaluative criteria. Qualitative Sociology **13**, 3–20 (1990)
25. Ntentos, E., Warnett, S.J., Zdun, U.: Supporting Architectural Decision Making on Training Strategies in Reinforcement Learning Architectures. In: 2024 IEEE 21st International Conference on Software Architecture (ICSA). pp. 90–100 (2024). https://doi.org/10.1109/ICSA59870.2024.00017

26. DeFranco, J., Laplante, P.: A content analysis process for qualitative software engineering research. Innovations in Systems and Software Engineering **13**, 1–13 (09 2017). https://doi.org/10.1007/s11334-017-0287-0
27. Warnett, S.J., Zdun, U.: On the Understandability of MLOps System Architectures. IEEE Transactions on Software Engineering **50**(5), 1015–1039 (2024). https://doi.org/10.1109/TSE.2024.3367488
28. Ntentos, E., Warnett, S.J., Zdun, U.: On the understandability of machine learning practices in deep learning and reinforcement learning based systems. Journal of Systems and Software **222**, 112343 (2025). https://doi.org/10.1016/j.jss.2025.112343, https://www.sciencedirect.com/science/article/pii/S0164121225000111
29. Warnett, S.J., Ntentos, E., Zdun, U.: A model-driven, metrics-based approach to assessing support for quality aspects in MLOps system architectures. Journal of Systems and Software **220**, 112257 (2025). https://doi.org/10.1016/j.jss.2024.112257, https://www.sciencedirect.com/science/article/pii/S0164121224003017
30. Ntentos, E., Warnett, S.J., Zdun, U.: Rule-Based Assessment of Reinforcement Learning Practices Using Large Language Models. In: 2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN). pp. 1–11 (2025). https://doi.org/10.1109/CAIN66642.2025.00009
31. Faubel, L., Schmid, K.: MLOps: A Multiple Case Study in Industry 4.0. In: 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA). pp. 01–08 (2024). https://doi.org/10.1109/ETFA61755.2024.10711136
32. Baldoni, M., Baroglio, C., Ditano, V., Micalizio, R., Tedeschi, S.: Agents for Industry 4.0: the Case Study of a Production Cell. In: Workshop From Objects to Agents (2023), https://api.semanticscholar.org/CorpusID:266211460
33. Kegyes, T., Süle, Z., Abonyi, J.: The Applicability of Reinforcement Learning Methods in the Development of Industry 4.0 Applications. Complexity **2021**(1), 7179374 (2021). https://doi.org/10.1155/2021/7179374
34. Übellacker, T.: AcademiaOS: Automating Grounded Theory Development in Qualitative Research with Large Language Models (2024), https://arxiv.org/abs/2403.08844
35. Yue, Y., Liu, D., Lv, Y., Hao, J., Cui, P.: A Practical Guide and Assessment on Using ChatGPT to Conduct Grounded Theory: Tutorial. J Med Internet Res **27**, e70122 (May 2025), https://doi.org/10.2196/70122
36. Apvrille, L., Sultan, B.: System Architects Are not Alone Anymore: Automatic System Modeling with AI. In: Mayo, F.J.D., Pires, L.F., Seidewitz, E. (eds.) Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering, MODELSWARD 2024, Rome, Italy, February 21-23, 2024. pp. 27–38. SCITEPRESS (2024)
37. Keshri, R., Zachariah, A., Boone, M.: Enhancing Code Consistency in AI Research with Large Language Models and Retrieval-Augmented Generation (02 2025). https://doi.org/10.48550/arXiv.2502.00611
38. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA (2020)