The Journal of Systems and Software 217 (2024) 112180

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

An architecture for model-based and intelligent automation in DevOps☆

Romina Eramo [a,*], Bilal Said [b], Marc Oriol [c], Hugo Bruneliere [d], Sergio Morales [e]

[a] Department of Communication Science, University of Teramo, Teramo, Italy
[b] Softeam Software BU, Docaposte, Nantes, France
[c] Universitat Politècnica de Catalunya, Barcelona, Spain
[d] IMT Atlantique, LS2N (UMR CNRS 6004), Nantes, France
[e] Universitat Oberta de Catalunya, Barcelona, Spain

ARTICLE INFO

Keywords:
Software architecture
DevOps
Continuous software engineering
Artificial Intelligence
Mode-driven engineering

ABSTRACT

The increasing complexity of modern systems poses numerous challenges at all stages of system development and operation. Continuous software and system engineering processes, e.g., DevOps, are increasingly adopted and spread across organizations. In parallel, many leading companies have begun to apply artificial intelligence (AI) principles and techniques, including Machine Learning (ML), to improve their products. However, there is no holistic approach that can support and enhance the growing challenges of DevOps. In this paper, we propose a software architecture that provides the foundations of a model-based framework for the development of AI-augmented solutions incorporating methods and tools for continuous software and system engineering and validation. The key characteristic of the proposed architecture is that it allows leveraging the advantages of both AI/ML and Model Driven Engineering (MDE) approaches and techniques in a DevOps context. This architecture has been designed, developed and applied in the context of the European large collaborative project named AIDOaRt. In this paper, we also report on the practical evaluation of this architecture. This evaluation is based on a significant set of technical solutions implemented and applied in the context of different real industrial case studies coming from the AIDOaRt project. Moreover, we analyze the collected results and discuss them according to both architectural and technical challenges we intend to tackle with the proposed architecture.

1. Introduction

Modern systems in Industry 4.0, in domains such as health care, autonomously driving cars, or smart grids, are examples of highly communicating (embedded) systems where software enables increasingly advanced functionality. The growing complexity of these systems poses several challenges throughout all the system design, development, and analysis phases, as well as during their deployment, actual usage, and future maintenance (Thompson et al., 2018; Eramo et al., 2022). Continuous software engineering (Fitzgerald and Stol, 2017) is a promising paradigm that interleaves business strategy (i.e., requirement engineering) with development and operations as a continuum. It aims to produce better software with more successful implementations while satisfying requirements and constraints.

Similarly, today's emphasis on DevOps (Ebert et al., 2016; Jabbari et al., 2016) confirms the fact that the integration between software development and its operational distribution needs to be continuous. Indeed, DevOps improves end-to-end collaboration between stakeholders, development, and operations teams. It also promotes faster iteration, shorter feedback loops, and increased automation for better-quality software delivery.

In parallel, many companies started to deploy Artificial Intelligence (AI) principles and techniques in some specific parts of their businesses (Gartner, 2019). By integrating AI into DevOps processes, organizations can leverage predictive analytics, automation, and intelligent decision-making to streamline workflows and improve operational efficiency. For example, it has been successfully used in disciplines such as security and testing (Leite et al., 2019). Nowadays, AI for IT operations (AIOps) (Charley Rich and Pankaj Prasad, 2019b) guides and automates operational tasks and may also ingest metrics and use inference models to extract actionable insights from data. AIOps will evolve in the coming years by increasingly supporting the DevOps pipeline through continuous monitoring, alerting, and remediation securely and reliably.

☆ Editor: Raffaela Mirandola.

* Corresponding author.
E-mail addresses: reramo@unite.it (R. Eramo), bilal.said@docaposte.fr (B. Said), marc.oriol@upc.edu (M. Oriol), hugo.bruneliere@imt-atlantique.fr (H. Bruneliere), smoralesg@uoc.edu (S. Morales).

https://doi.org/10.1016/j.jss.2024.112180
Received 22 December 2023; Received in revised form 4 May 2024; Accepted 30 July 2024
Available online 2 August 2024

Although DevOps technology has seen significant advancements in recent years, organizations often have "local" implementations per team or development phase, considering the DevOps pipeline separately. Moreover, as DevOps and AI technologies mature and evolve, organizations must be ready to embrace these changes. Thus, organizations need to rely on a holistic approach that can support and enhance the DevOps process and the growing challenges of continuously developing modern complex systems (Leite et al., 2019).

We argue that the continuous engineering of complex systems, particularly in DevOps processes, requires dedicated AI-augmented methods and tools to extract knowledge from event streams (e.g., real-time and historical data) and design information (e.g., different system models). The objective is to extract meaningful insights for system improvement, drive faster deployments, and foster better collaboration, and reduce downtime with proactive detection. In this context, Model Driven Engineering (MDE) (Schmidt, 2006; Brambilla et al., 2017) is a relevant Software Engineering paradigm to raise the abstraction level and improve the ability to handle complexity. The use of models as first-class abstractions of systems and their environments is a fundamental element for technologies in current and future software/system engineering platforms (Thompson et al., 2018; Combemale and Wimmer, 2019). Moreover, it is worth noting that there is still a gap in the literature regarding contributions that explicitly propose or handle the combination of principles, practices, and tools related to MDE, DevOps, and AI. Indeed, when studying the state-of-the-art (Berardinelli et al., 2022), we observed that combinations of two of these three areas are already relatively frequent (e.g., MDE and DevOps). However, few approaches actually intend to combine MDE, DevOps and AI altogether in order to achieve common objectives. Moreover, when they do, they are not systematically featured with a dedicated tooling support.

In this paper, we introduce a software architecture for a model-based framework incorporating methods and tools for continuous software/system engineering and validation, leveraging the advantages of AI techniques including Machine Learning (ML). Notably, the architecture target benefits significantly improved productivity, quality, and predictability of large and complex industrial systems.

This work is developed and demonstrated within the European AIDOaRt project[1] (Bruneliere et al., 2022; Eramo et al., 2021). The project aims to support systems engineering and continuous delivery activities, namely requirements engineering, modeling, coding, testing, deployment, and monitoring, with AI-augmented, automated MDE and development operations. The proposed architecture has been instantiated in practice, both by the project's academic and industrial partners implementing specific capabilities. Furthermore, these capabilities have been applied in 10 industrial case studies while developing and validating the previously mentioned features. Thus, we report on the effectiveness of the application and integration of the AIDOaRt architecture and framework within the case studies. We also discuss the main lessons we learned and the insights we gained.

The rest of this paper is organized as follows. Section 2 introduces the underlying background of the work. Section 3 illustrates the architecture while Sections 4 and 5 describe its integration strategy and its implementation, respectively. Section 6 evaluates the architecture based on its implementation and application in industrial case studies, Section 7 presents the threats to validity. In contrast, Section 8 discusses our main observations and lessons learned. Finally, Sections 9 and 10 present the related work and conclusion, respectively.

2. Background

This section introduces our work's underlying concepts, challenges, and context.

[1] AIDOaRt ECSEL-JU project: https://www.aidoart.eu/.

2.1. Basic concepts

Model Driven Engineering (MDE). MDE allows raising the level of abstraction and thus improving the ability to engineer and handle complex and software-intensive systems (Thompson et al., 2018). The use of models as purposeful abstractions of systems and environments is also increasing within the industry (e.g., digital twining Bordeleau et al., 2020b). While first-generation MDE tools mainly focus on generating code from high-level models, they now also address model-based testing, verification, measurement, tool/language interoperability, or software evolution, among many other software engineering challenges. Major MDE benefits include: - providing better abstraction principles and techniques (e.g., for the handled data), - facilitating the automation of engineering activities, and - supporting technology integration among all the covered design and development activities.

Artificial Intelligence and Machine Learning (AI/ML). The dissemination of Artificial Intelligence (AI), including Machine Learning (ML), principles and techniques in a regulated industry enables systems to decide and act in a more and more automated manner: it is used by companies to exploit the information they collect to improve the products and/or services they offer (Gartner, 2019). Lately, AI/ML is also impacting all aspects of the system and software development lifecycle, from specification to design, testing, deployment, and maintenance, with the main goal of helping engineers produce systems and software faster and with better quality while being able to handle ever more complex systems and software (Wan et al., 2021; Burgueño et al., 2021; Felderer et al., 2023). Numerous ML techniques have been applied within the AIDOaRt project, from supervised and unsupervised learning ML algorithms to deep learning techniques (Bonaccorso, 2017; Zhou, 2021; Albawi et al., 2017). For example, Heterogeneous Graph Neural Networks (HGNNs) have been applied for model inference purposes, Density-Based Spatial Clustering of Application with Noise (DBSCAN) for detect failure logs, and automata learning for testing (Learning-Based Testing - LBT). Further example includes Generative AI for test scenario generation or Large Language Model (LLM) for formally verify functional requirements (AIDOaRt Consortium, 2022b, 2023a, 2024). Overall, the solution providers freely chose the AI/ML techniques to adopt in their respective context. The choice was notably based on the requirements coming from the industrial case studies (AIDOaRt Consortium, 2022d, 2023b).

DevOps and AIOps. DevOps is a software engineering paradigm focused on software delivery by enabling continuous feedback and quick response to changes and using automated delivery pipelines resulting in reduced cycle time. In particular, tools and methods focus on administration and automation processes (Leite et al., 2019; Jabbari et al., 2016; Combemale and Wimmer, 2019). AIOps (Charley Rich and Pankaj Prasad, 2019a; Manjunath Bhat, 2019) characterize solutions where DevOps challenges are addressed with the help of AI/ML techniques. A fundamental challenge notably resides in integrating AIOps (Dang et al., 2019) in enhancing the DevOps pipeline, e.g., through continuous alerts and insights from data used for continuous deployment and operations management.

In this work, we propose a software architecture for a model-based framework incorporating methods and tools for continuous software/system engineering and validation (within DevOps), leveraging the advantages of AI/ML and MDE principles and techniques.

2.2. Challenges

As a result of our study of the state-of-the-art in terms of combining MDE, DevOps and/or AI (Berardinelli et al., 2022), we identified a large variety of existing approaches. These approaches are often quite generic concerning the potential target domains. Moreover, they also cover several major software engineering activities (e.g., requirements, modeling, coding, testing, monitoring). In this respect, these

existing approaches intend to address different main challenges, both architectural and more technical ones.

In this paper, we want to highlight the importance of supporting some of the most commonly encountered architectural challenges related to the continuous development (i.e., DevOps) of large and complex software-intensive systems (e.g., CPSs). Among the architectural challenges frequently identified within the literature in this context (Hofer, 2018; Törngren and Sellgren, 2018; Zampetti et al., 2022), we notably consider the following:

- *A1. Heterogeneity*. The continuous development (i.e., DevOps) of the nowadays complex and heterogeneous systems (i.e., CPSs) require to be supported at different levels of the architecture. Having different domains and case study scenarios of these complex systems implies having heterogeneous requirements, several functional capabilities, and different data models/formats.
- *A2. Flexibility*. Considering the different requirements arising from the numerous CPS scenarios and case study, it is necessary to provide, through the architectural model, more than one option for the same feature or provided functionality. The model must allow the options to be chosen, considering the advantages and disadvantages of each one. Flexibility in architectural instances is related to the different implementations of the components and interfaces.
- *A3. Integration*. This challenge concerns integrating multiple components to connect solutions, tools, data, models, and services implemented within the framework. It also enables the integration of the different phases of DevOps as supported by the architecture.
- *A4. Collaboration*. Collaboration is a cornerstone of DevOps. It involves development, operations, and different stakeholders. Architecture plays a fundamental role in supporting that collaboration, enabling effective communication, continuous feedback, and sharing knowledge and solutions/tools.

In addition to these architectural challenges, we also identified the following technical challenges as particularly important (among others well-known in the literature Russo et al., 2020; Törngren and Sellgren, 2018; Derler et al., 2012; Bergelin and Strandberg, 2022) for the continuous development of such complex systems.

- *T1. Data management*. With the increasing complexity of systems, the amount of generated data increases exponentially. This data must be properly managed, stored, and analyzed to extract meaningful insights.
- *T2. Modeling support*. This challenge is related to using models as the primary artifact of the development process, enabling stakeholders to communicate, analyze, and refine system requirements and design decisions.
- *T3. AI-powered DevOps*. Integrating AI into the development process brings many possibilities, such as increased efficiency, improved decision-making, and the ability to operate autonomously. AI is expected to enable control, monitoring, testing, management, optimization, prediction, and automation.

Overall, both these architectural and technical challenges are fundamental objectives we considered we elaborating on the architecture we propose in this paper (cf. Section 3). They have also been used as main criteria for evaluating this architecture (cf. Section 6), with the direct support from our industrial partners in the AIDOaRt project.

2.3. The AIDOaRt project

The AIDOaRt project (Bruneliere et al., 2022) aims at supporting systems engineering and continuous delivery activities, namely requirements engineering, modeling, coding, testing, deployment, and monitoring, with AI-augmented, automated MDE and DevOps.

To achieve this goal, AIDOaRt proposes a model-based architecture that specifies proper methods and tools to enable design and run-time data collection, ingestion, and analysis to provide tailored and efficient AI/ML solutions. These solutions are then integrated and evaluated on concrete industrial case studies involving various Cyber-Physical Systems (CPSs).

AIDOaRt is rich in the number of partners and the variety of case study requirements and solutions capabilities, demanding a real need for a common architecture to enable collaboration between the various partners. Thus, it also requires adopting a rigorous and agile methodology to formally model and define the architecture, incrementally collect, refine, and clarify the requirements, and map them to the candidate solutions.

To this end, we adopted a Model-based Requirements Engineering (MBRE) approach (Sadovykh et al., 2021, 2024) that we applied in several incremental iterations (Said et al., 2022). The adopted agile methodology was implemented using the distributed model repository Modelio SaaS[2] (Desfray, 2015) In practice, a global model was created and shared with the consortium partners as a single source of truth. As the work in the project progresses, discussions and in-depth negotiations occur between case study providers and solution providers. This leads to precise and concrete case studies and data requirements, as well as to better choices and selections of candidate solutions. To report on such refinements, the core team defines sets of new *micro-tasks* (similar to sprints in agile processes), requesting the partners to provide new model elements. This allowed for the timely integration of frequent updates on requirements and solution descriptions from all the partners in a flexible, collaborative, and traceable way.

This model-based and agile approach also allowed us to gradually specify our AIDOaRt architecture by refining it at each project milestone while keeping the case study requirements and solution descriptions aligned with this architecture specification. Thus, the architecture provided a common ground for integrating the AIDOaRt solutions into the case studies and identifying potential collaborations between solution providers and case study providers. This notably prevents each partner from having to systematically check long lists of requirements and solutions.

3. Proposed software architecture

The AIDOaRt architecture is specified in terms of engineering activities that combine principles and practices from MDE and DevOps, augmented by Artificial Intelligence and Machine Learning (AI/ML) optimizations and enhancements. The initial version of the proposed architecture has been defined based on our previous academic and industrial experience in similar contexts. Notably, we started from our previous work on providing a global architecture for the continuous development and runtime validation of complex systems in the context of the MegaM@Rt2 project (Afzal et al., 2018). However, while relying on MDE and some DevOps practices, the architecture proposed in MegaM@Rt2 was not natively integrating the use of AI/ML as one of its core element. Thus, we capitalized on the MegaM@Rt2 results and on other architectural experiences within various collaborative projects (Sadovykh et al., 2021, 2024) in order to iterate several times on the AIDOaRt architecture. The version presented in what follows is the final result of the application of such an iterative process.

Contrary to MegaM@Rt2 whose scope was more limited, AIDOaRt aims at supporting the major software and systems engineering activities in a real DevOps setting. As shown in Fig. 1, these activities are requirements engineering, modeling, coding, testing, operation, monitoring, and feedback.

[2] More on Modelio SaaS on https://www.modeliosoft.com/ and Modelio community version on https://www.modelio.org/.

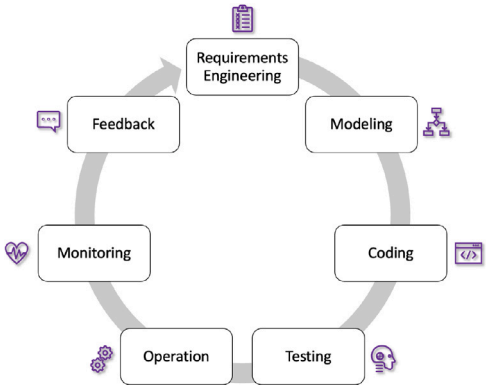

Fig. 1. Considered DevOps process.

We define the architecture as a hierarchy of UML *components* and sub-components, organized in three *layers* (cf. Fig. 2), and offering/consuming services and capabilities defined as *functional interfaces*. Notably, we identified components and interfaces related to the AI-augmented capabilities for the various software engineering (SE) activities or DevOps phases: Requirements Engineering, Modeling, Coding, Testing, and Monitoring (Bruneliere et al., 2020; Maalej et al., 2016; Valente et al., 2021; Sebastián et al., 2020). At this level, we could trace a first integration link between the architecture and the partners' case studies and solutions capabilities through their relations to the common generic requirements.

To enable the implementation and integration of the above-mentioned AI-augmented SE and DevOps capabilities, the architecture captures a set of common AIOps Engineering capabilities (Manjunath Bhat, 2019), such as Ingestion, Data Handling, Engagement, Analysis, and Automation. These AIOps Engineering capabilities rely on a set of Core Engineering capabilities such as Data, MDE, Storage, and Computation, as well as Accountability and Explainability services (Barredo Arrieta et al., 2020; Blumreiter et al., 2019). Finally, these AIOps and Core Engineering capabilities rely on a set of data-centric capabilities, namely Data Collection, Management, and Representation, to provide the required data-driven AI services.

Fig. 3 shows the mapping between the considered DevOps phases (in Fig. 1) and the AIDOaRt architecture components.

In what follows, we dive deeper into the specification of each component in each of the three layers of the AIDOaRt architecture, namely the Data Engineering, Core, and AI-augmented toolsets.

3.1. Data engineering tool set

This layer provides the required capabilities to support the collection, management, and representation of data. Collected data conform to defined metamodels following MDE principles and are provided to the *AI-Augmented Tool Set* through the *Core Tool Set* capabilities.

The **Data Collection** sub-component intends to support data collection processes from different data sources at both run-time and design-time.

Data Management supports cleaning, analyzing, and managing data coming from different sources. It is designed to provide capabilities for (1) *filtering and harmonization* of data collected from various design and run-time sources, (2) *data transformation* into unified internal representation as defined by the **Data Representation** component, as well as (3) *data aggregation* capabilities to combine various data sources into coherent processable data blocks and streams.

The **Data Representation** component provides a common, agreed-upon, global data representation to serve as the foundation for MDE-based activities throughout the AIDOaRt framework. To achieve this, it leverages a **mega-model** that unifies the different case study data models by normalizing their notation and linking those concepts that are (partially) synonymous across the various concrete data models. It yields generic representations for those elements that are shared across the different specific data models, thus providing an agnostic overview of the most relevant data elements present in the AIDOaRt framework and its development process.

Fig. 4 illustrates the two-layer structure of the mega-model: the "blue" layer depicts the different case study data models, whereas the "orange" layer portrays the abstract elements that generalize the specific shared concepts and links them.

The generic "orange" layer of the mega-model is divided into packages, each of them representing the data related to a stage of the AIDOaRt process: Requirements Engineering, Modeling, Coding, Testing, and Monitoring (cf. Fig. 5). The *Requirements Engineering data model* contains generic elements to represent the functional and non-functional requirements of a system, derived from an analysis of the stakeholders and context needs. The *Modeling data model* provides an abstract representation of a system, an element of a system, or its behavior. The *Coding data model* includes elements that abstract the coding process and its main artifacts. The *Testing data model* contains constructs for representing the definition and automation of tests, and the adapting nature of the AI models (and other physical or logical elements) being tested. Finally, the *Monitoring data model* encompasses the data that is collected from the sensors and the measurements performed against a set of evaluation metrics to, whenever it is possible, automatically assess if an anomaly or degradation of the system has occurred. Those metrics are set according to the defined target performance of the system. It also provides elements to represent logs collected throughout the system, throughout all its development phases.

Artifacts from all packages could be interconnected, as they are shared across different stages of the development process. In its final implementation, some *Testing data model* elements are linked to artifacts from the *Requirements Engineering data model* and the *Monitoring data model*, hence its "*use*" relationships with both packages.

The goals of such a mega-model are many-fold. First, by adopting a common language for all case study data models, namely the UML notation, we homogenized how data and their structure are designed. A standard notation enables a common understanding of the concepts, and fosters communication and collaboration between the users, thus facilitating better knowledge sharing. We adopted UML for data representation because of its ability to perform formal verification of data models and the wide support from solution providers.

Second, the mega-model can be used to unveil synergies between case studies, thus identifying further potential collaborations. Given a generic element that abstracts two concrete concepts from different case study domains, stakeholders from one domain could establish a collaboration with stakeholders from the other connected domain. In this partnership, they can share their expertise, enrich their respective data models, and apply similar strategies, tools, or techniques to common problems. Furthermore, this approach could also allow solution providers to better comprehend the transversal application of their tools in different scenarios.

The generic layer of the mega-model provides users with a bird's eye view of the main concepts managed across the AIDOaRt case studies. We expect it to facilitate the onboarding of new users of the framework, who could easily grasp the generic concepts and their relationships, and, if interested, dive deep into a specific domain using the links of the generic elements to the concrete ones.

3.2. Core tool set

This layer provides generic capabilities that are intended to be used by the other toolsets. It includes the global infrastructure, shared data, and modeling features, as well as common generic services, as detailed in the sequel.

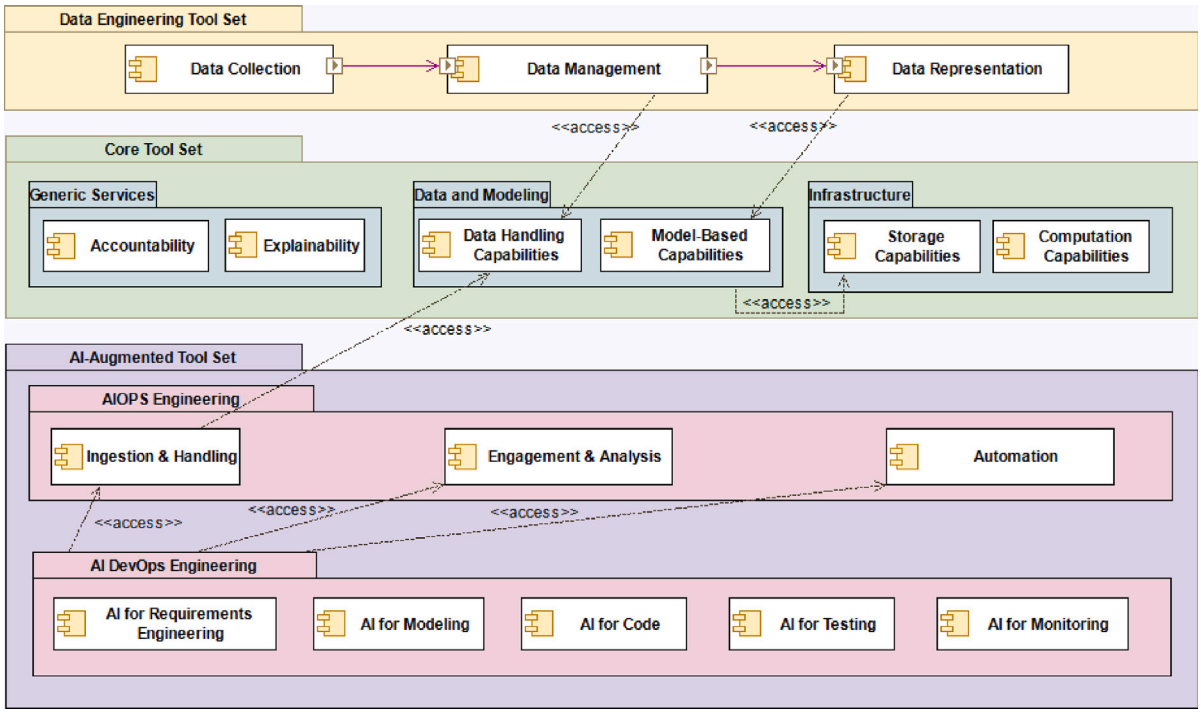

Fig. 2. Overview of the AIDOaRt architecture.

DevOps Phases	AIDOaRt Architecture Components
Requirements Engineering	AI for Requirement Engineering
Modeling	Data Representation, Model-based capability, AI for Modelling
Coding	AI for Code,
Testing	AI for Testing
Operation	Data management, Data Handling capability, Storage capability, Computation capability, Ingestion & Handling, Automation
Monitoring	Data collection, AI for Monitoring
Feedback	Engagement & Analysis, Accountability, Explanability

Fig. 3. Mapping between the considered DevOps phases and the AIDOaRt architecture components.

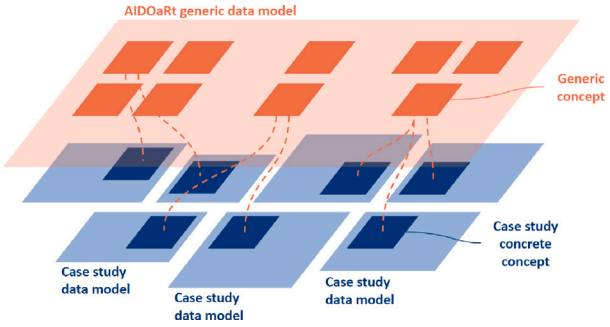

Fig. 4. A representation of the two layers of the AIDOaRt mega-model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

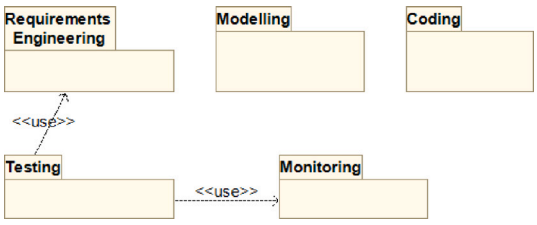

Fig. 5. The AIDOaRt mega-model packages.

The **Infrastructure** components support basic infrastructure capabilities in terms of data and model storage and related computations. This support can be implemented by open technical platforms (e.g., public repositories or computation grids) or by companies' infrastructures (e.g., in the context of AIDOaRt case studies). On one hand, **Storage Capabilities** support the provisioning of physical and logical resources. Solutions implementing these capabilities are expected to efficiently store and retrieve the possibly numerous and large software artifacts, such as data and models, that are hosted on various media sources, e.g., repositories, remote file systems, and databases. On the other hand, the **Computation Capabilities** support the provisioning of physical and logical resources to process and analyze these artifacts. Solutions implementing these capabilities may consume such artifacts as inputs of computations or restitute back these computations results.

The **Data and Modeling** components support general data and model handling capabilities, which may be implemented by generic and/or open-source solutions (e.g., Eclipse/EMF-based tools) or by the company's specific solutions (e.g., in the context of AIDOaRt case studies).

More in detail, **Data Handling Capabilities** support (1) *loading* different kinds of data in-memory for further manipulation and/or

treatment, using different formats of serialization or representation, (2) *navigating* and browsing the data and its elements once loaded, easily and efficiently, (3) *querying* the data to obtain specific elements easily and efficiently, and then (4) *saving*, in diverse serialization or representation formats, data residing in memory to permanent logical and physical storage locations and media.

Similarly, **Model-Based Capabilities** support loading, navigation, querying, and saving of models and metamodels. In contrast, they also support *transformation* of a loaded model into another model, *tracing/federation* (e.g., via corresponding views) different models and their elements altogether, as well as *code generation* to generate source code (or any kind of structured text) from a model. In addition, we recommend that solutions implementing this component would also provide *Modeling Process Tracing* capabilities, which intend to collect user interaction during modeling activities. This allows for a variety of AI/ML augmented modeling solutions to be developed by learning from expert modeling activities to assist other modelers with relevant insights (see next layer *AI-Augmented Tool Set*).

Finally, we specified a last core component called **Generic Services**. It intends to support services relevant in an AI/ML context, such as (but not limited to) the **Accountability** and **Explainability**, which provide generic management of responsibility and understanding of the results obtained via the used AI/ML techniques. Solutions implementing these services are expected to analyze software artifacts and deliver the analysis results by relying on or exploiting the approaches and techniques provided by the *AI-Augmented Tool Set* that is presented next.

3.3. AI-augmented tool set

This layer supports the development of the AI-augmented toolchain that extends the *Core Tool Set* components according to the needs of the various AIDOaRt case study requirements. Its **AIOps Engineering** component offers capabilities related to the AIOps methodology by supporting the ingestion of data, events, and metrics from various sources, engagement and analysis by employing AI techniques, and the automation of operations belonging to the development process. Whereas the **AI DevOps Engineering** component offers additional capabilities related to the different system engineering and DevOps tasks. This includes the application of AI for requirements engineering, modeling, coding, testing, and monitoring, allowing a combination of AIOps and DevOps in MDE settings.

Accordingly, **Ingestion and Handling** is designed to support the *continuous monitoring* of design/run-time data, such as historical and real-time data acquisition and collection, as well as visualization and query capabilities, for instance using dashboards and API calls. It also supports capabilities to *filter* through massive, redundant, or noisy amounts of data. In addition, it may provide ML-based techniques for *dataset balancing*, e.g., oversampling, under-sampling, class weight, and decision threshold. This component also provides *pattern discovery* capabilities to correlate, contextualize, and find relations between meaningful data elements to group them for advanced analytics. Furthermore, it provides *Bug Pattern Discovery* capabilities that use AI/ML methods for automated learning of patterns to detect (functional or performance) bugs in networks and embedded systems. Last but not least, it supports *API and Code Patterns Discovery* capabilities to capture strategies for pattern discovery, e.g., collaborative filtering for analyzing OSS projects or capabilities to find relevant API code.

On the other hand, we group under the **Engagement and Analysis** component the analysis capabilities (inference, prediction, deduction, etc.), as well as their collaboration and integration. *Insights analysis* capabilities intend to offer root cause analysis, inference, deduction, verification, consistency checks, and design space exploration based on AI/ML techniques. Whereas *predictive analysis* capabilities regroup AI/ML-based prediction (e.g., of potential undesired scenarios) using AI/ML techniques, algorithms, and models trained and fed with different sources of data (e.g., models). *Collaboration services* denote capabilities allowing notification among components and integration of services to ensure that any incidents, dependencies, and changes made across the environments are synced with the whole architecture toolset.

In addition, we specify an extra subset of AI/ML-based capabilities that are key in providing engagement and analysis for AIOps Engineering. Namely, we specify *AI/ML for Anomaly Detection* capabilities to represent algorithms and approaches for the detection of anomalies in time series monitoring data to effect system changes. We also specify *ML-based Prediction for Human–Machine Interaction (HMI)* to capture ML-based analysis and prediction capabilities for HMI in CPS (e.g., prediction of human driver behavior in automotive systems). We also capture *ML-based Object Detection* capabilities that provide automated and intelligent analysis and object detection in images and video streams (e.g., videos captured by AI-augmented self-driving cars). With *AI for Equivalence Class Prediction*, we capture AI techniques for the prediction of equivalence classes (EC) based on previously used ECs in the test model. Moreover, we specify *ML-based Prediction For Performance and Resource Utilization* to capture AI/ML analysis and estimation for hardware and/or software platforms timing performance (e.g., response time, execution time, hardware latency) and platform resource utilization (e.g., FPGA area usage, memory allocation, etc.).

Finally, the **Automation** component of AIOps Engineering supports the codification of the obtained insights into remediation and response automation. *Remediation* capabilities are needed to leverage prescriptive advice (e.g., pattern-based prediction) and to take automated action (i.e., remediation, proactive operation) to support system development. *Response automation* represents automated mechanisms for efficient coding of human domain knowledge obtained from remediation into response automation and orchestration (e.g., automate routine tasks, incident handling, threat mitigation).

The second major component in the *AI-Augmented Tool Set* layer is **AI DevOps Engineering**. This component offers AI/ML augmented capabilities to support the various phases of systems engineering and DevOps; namely, requirements engineering, modeling, coding, testing, and monitoring.

The **AI for Requirements Engineering** sub-component enables the integration of AI/ML techniques to support the requirements engineering phase of systems development. This includes NLP techniques for textual requirements *Ambiguity Check*, *Semantic Similarity Check*, *Classification* (i.e., clustering into categories, e.g., functional/non-functional, usability, performance, security, etc.) and *Allocation* (i.e., auto-assignment to departments, teams or individual engineers). These capabilities may consist of ML models trained on labeled corpora from previous projects, or trained with zero-shot learning, few-shots learning, or reinforcement learning from human feedback (RLHF) with the help of expert requirement engineers. Furthermore, we propose to capture AI/ML capabilities for *Model Consistency Verification*, such as automated formal methods for consistency verification of system design models, *Specifications Consistency Verification*, e.g., formal methods, automated reasoning, and NLP techniques for the automated consistency verification of technical specifications w.r.t. guidelines or project-specific criteria, and *Reuse Analysis and Recommendation*, e.g., AI/ML and NLP techniques to identify similar assets across previous projects and recommend their reuse in new ones.

We designed the **AI for Modeling** sub-component to support the modeling phase of the system development with AI/ML techniques for various modeling activities. For instance, we expect solutions and tools implementing this component of the architecture to leverage AI/ML techniques for *Modeling Assistance*, e.g., to use Graph Neural Networks (GNN) to assist modelers with relevant insights and recommendations, *Design Space Exploration*, e.g., AI-driven optimization algorithms for design space exploration, *View-Model Synchronization*, e.g., GNN used to automate view-model synchronization, *Model Learning*, e.g., methods for extracting models from existing systems or data, and *Instance Model Generation*, e.g., AI/ML for generating instance models of a given DSL.

The **AI for Code** sub-component is intended to enable the use of AI/ML techniques to support the coding phase of the system development. For instance, it provides capabilities for *Recommending Code Snippets*, e.g., collaborative filtering techniques used to recommend API calls automatically, *Functional Code Generation*, e.g., for C/C++/SystemC functional code generation starting from high-level system specification, and *Factory Methods and Mocking*, e.g., learning from user-written factory methods to generate mocking code.

For the **AI for Testing** sub-component, we identified the need to capture AI/ML techniques for automatic test generation, such as *AI for Test Suite Generation* and *AI for Unit Test Generation*, as well as *AI for Test Model Generation* which denotes AI techniques for the analysis and transformation of test models. In addition, we also identified the need to provide AI/ML capabilities for *Learning Based Testing*, e.g., intelligent testing techniques based on models and requirements learning, and for *Test Case Reduction*, e.g., automated techniques for test case reduction of an equivalence set based test model.

Among the multitude of AI/ML capabilities that we specify in our **AI for Monitoring** sub-component to enhance and support systems monitoring at real-time and run-time, we identify *AI for Text Analytics*, which targets the use of AI to analyze large volumes of semi-structured text (e.g., abundantly generated system logs), *AI/ML based Functional/Performance Bug Detection*, namely AI/ML techniques for detecting out of the monitored data (on virtual model) functional or performance bugs, and *AI/ML for Anomaly Detection*, which groups AI/ML algorithms for the detection of anomalies in time series monitoring data (offline or online) related to system/network's bugs or malfunctions.

3.4. Cross-layer components integration

Our architecture components do not live in isolated silos. On the contrary, they are meant to be aggregated and integrated through complex data and control flows to achieve more complex functional behaviors. Furthermore, component correlations provide insights to case studies and solutions providers about potential horizontal and vertical integration aspects of their systems and tools. For example, as shown with the "access" relations in Fig. 2, the **Data Management** and **Ingestion & Handling** components rely on the **Data Handling Capabilities** component to load, discover (i.e., navigate, query, select, filter, aggregate) and persist data. This also applies to solutions and tools implementing these components. For instance, Fig. 6 shows that partners UNIVAQ and DT can potentially reuse in their respective tools "Keptn" and "HEPSYCODE" some data handling capabilities offered by the "DataAggregato" solution proposed by partner ROTECH. This is an example of a potential horizontal integration between solutions. The same applies to horizontal integration between two or more case study scenarios or systems. Similarly, vertical integration might be established between case studies and solutions based on their relationship to the architecture components. In Section 4, we provide more details on these integration aspects.

Similarly, the **Data Representation** component relies on **Model-based Capabilities** to define and manage data models and metamodels. Both **Data and Modeling** components use **Storage Capabilities** to efficiently store and retrieve potentially numerous and large data and model artifacts. It is also natural for any **AI DevOps Engineering** component to access the **Ingestion & Handling**, **Engagement & Analysis**, and **Automation** components to continuously monitor data sources, analyze the collected data, and transform the obtained analysis insights into remediation and response automation, respectively.

The following sections illustrate the integration strategy of the presented architecture and how it has been concretely implemented and evaluated in the context of the AIDOaRt project.

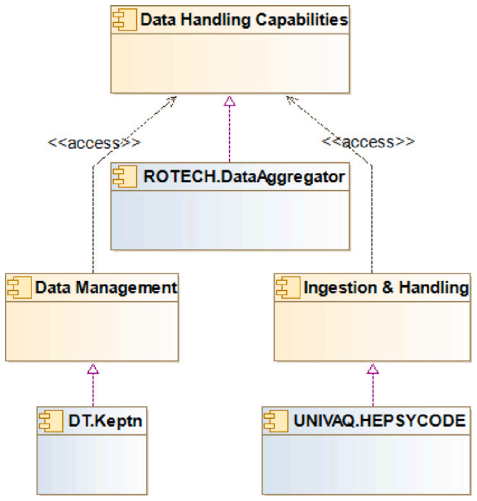

Fig. 6. Horizontal integration between solutions based on the integration of their respective architecture components.

4. Integration strategy

The main aim of the AIDOaRt architecture is to provide a common ground for the integration of the AIDOaRt solutions into case studies and the identification of potential collaborations between solution and case study providers.

To perform these integration activities in an effective and systematic way, we designed a set of integration strategies that facilitates the discovery of potential relationships between case study requirements and solutions' capabilities. These integration strategies have been formalized as *integration mediation patterns*. They are generic patterns that relate an aspect of a case study (e.g., case study scenario, functional requirement, data requirement) with an aspect of a solution (e.g., component, functional interface) following a specific strategy. We defined four different types of integration mediation patterns in order to cover our needs in the project.

4.1. Generic requirements mediation pattern

This mediation pattern deals with the integration by generalizing case study requirements related to the different AI-supported software development phases. First, high-level generic requirements are elicited by grouping and generalizing case study-specific requirements. For instance, multiple case studies may require the use of AI to streamline their system configuration, each of them with their own particular needs. All these case study requirements could be generalized to a single generic requirement "use of AI-based methods for easy configuration". Iteratively, new case study requirements can be added to either (1) existing generalized requirements or (2) a new high-level generic requirement if needed. These generalized requirements are then categorized into the AI-supported software development phase they correspond to, i.e., requirements for AI for Requirements Engineering, requirements for AI for Modeling, requirements for AI for Code, requirements for AI for Testing, and requirements for AI for Monitoring. From our previous example, it would be mapped to AI for Modeling. Afterwards, solution providers explore these generic requirements for the AI-supported software development phase(s) of their interest and map their solutions to the generic requirements they are able to solve. This way, they create indirect links to specific case study requirements and discover potential relationships with case studies. In our example, if a solution provider possesses an AI tool for configuration support, they can map their solution to the generic requirement and examine the specific requirements of individual case studies.

4.2. AIDOaRt components mediation pattern

This mediation pattern deals with the integration through the components of the AIDOaRt architecture. Case study providers map their specific requirements to the components of the AIDOaRt architecture that are needed to implement and solve each requirement. For instance, a case study provider may require a Quality of Service (QoS) monitor to collect data for continuous AI training. This requirement can be mapped to the Data Collection component within the AIDOaRt architecture In parallel, solution providers map their solutions to the components of the AIDOaRt architecture they provide an implementation for. For instance, a solution provider having a QoS monitor, would map their solution to the Data Collection component of the AIDOaRt architecture. Afterwards, both case study and solution providers can refine their mappings to the particular interfaces of the AIDOaRt architecture's components. As a result, indirect links between case study requirements and solutions are established.

4.3. Data engineering mediation pattern

Data representation drives the concrete integration of this pattern. Case study providers identify the data representations related to their data requirements (i.e. their case study data models). These data representations are grouped by the domain they address, and generalized in a similar approach as in the generalization of requirements and finally analyzed to identify the potential synonyms or overlapping concepts. The result is the AIDOaRt mega-model, which unifies the different case study data models by normalizing their notation and linking those concepts that are (partially) synonymous across the various concrete data models. It yields generic representations for those elements that are shared across the specific data models, thus providing an agnostic overview of the most relevant data elements. This approach standardizes the terms across multiple domains and creates indirect links from a data perspective that uncovers potential relationships driven by data models. Further details and examples of how the AIDOaRt mega-model is built are presented in Section 5.2.

4.4. Case study's environment extensions

While the other integration patterns focus on integration aspects from the case study providers to the solution providers, this pattern follows a solution provider-driven integration approach. To this end, case study environments are presented as required enhancements for a successful integration. Such an integration is realized by coordinating possible synergies and collaboration with solution providers experts in the relevant area(s). For instance, in the context of the AIDOaRt project, this can be achieved by means of hackathons presenting challenges going beyond the defined case study requirements.

Each interested partner can use the most convenient pattern according to their integration needs. For example, case study providers can use AIDOaRt to discover in a systematic manner potential solutions that might be able to satisfy their requirements. Solution providers can identify, through AIDOaRt, potential case studies where they can deploy their solutions. At the current stage of the project (Eramo et al., 2023), we already conducted the integration of case study providers and solution providers following the *AIDOaRt components mediation pattern*, and the *Generic Requirements mediation pattern*.

For the*AIDOaRt components mediation pattern*, we proceeded as follows: (1) solution providers conducted a mapping of their solutions to the architecture components' capabilities and functional interfaces, and (2) case study providers conducted a mapping of the architecture components to their case study requirements and data requirements. From these two mappings, we automatically inferred the indirect links between the AIDOaRt solutions and the case study requirements. For the *Generic Requirements mediation pattern*, we defined 5 working groups (one per each AI-supported engineering phase). Each group was responsible for eliciting generic requirements from the specific case study requirements, and conducting the mapping as described before. Then, solution providers mapped their solutions to these generic requirements.

5. Implementation

The AIDOaRt architecture can be implemented within a broad range of projects (i.e., can be realized using different technologies, platforms, and frameworks) related to the development of complex systems. In this section, we discuss how the AIDOaRt components have been implemented in the context of the AIDOaRt project, and present the AIDOaRt mega-model as the main asset of the Data Engineering component.

5.1. AIDOaRt components implementation

The architecture presented above has been concretely implemented in the context of the AIDOaRt project. To this end, the features of each one of the components from the three Tool Sets (described in Section 3) are supported by individual technological solutions developed by the solution providers of the project (both academic and industrial partners). Each developed solution implements one or more architecture components from one or more of the three Tool Sets. At the same time, each component of the architecture can be implemented by one or more tools.

Within the AIDOaRt project, the architecture has been implemented by a large number of solutions (overall 136), covering the various components and capabilities as well as the generic requirements and specific requirements from the case study providers.

For dissemination and usage purposes, each one of the identified solutions has also been described in various AIDOaRt project deliverables[3] (cf. D2.1 AIDOaRt Consortium, 2022c, D2.2 AIDOaRt Consortium, 2022a, D2.3 AIDOaRt Consortium, 2023c, D3.2 AIDOaRt Consortium, 2022e, D3.3 AIDOaRt Consortium, 2022f, D3.4 AIDOaRt Consortium, 2023d, D4.1 AIDOaRt Consortium, 2022b, D4.2 AIDOaRt Consortium, 2023a, at the time of writing, and D4.3 AIDOaRt Consortium, 2024 to appear soon). Moreover, the case studies have also been described in specific AIDOaRt deliverables (cf. D5.5 AIDOaRt Consortium, 2022d for an excerpt, since the more detailed D1.1, D1.3, and D5.6 are confidential).

As a particular example, Fig. 7 shows how the Data Representation component from the AIDOaRt Data Engineering Tool Set is currently underpinned. For instance, EMF Views helps to integrate data from diverse data sources using different data model views. Another solution implementing this component is the AsyncAPI toolkit, which embeds a metamodel to effectively model data for message-driven platforms.

Furthermore, Fig. 7 shows the potential utility of this component and its related solutions in addressing, at least to some extent, several case study requirements regarding the need for data representation (cf. the *satisfy* links). Similarly, Figs. 8–10 depict the applicability of technical solutions and the satisfied requirements for the Model-Based, Engagement & Analysis and AI for Modeling capabilities components, respectively.

The number of solutions implemented for each component and some examples are described in Section 6.

5.2. AIDOaRt mega-model

At the basis of data engineering, the *Data Representation* component relies on the mega-model of the AIDOaRt framework. It is the result of

[3] https://www.aidoart.eu/aidoart/results/deliverables.

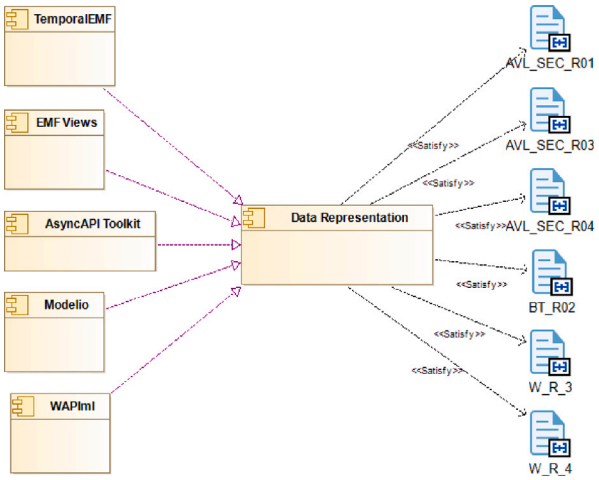

Fig. 7. Data Representation Capabilities component from the Data Engineering Tool Set, corresponding solutions and potential links with AIDOaRt case study scenarios.

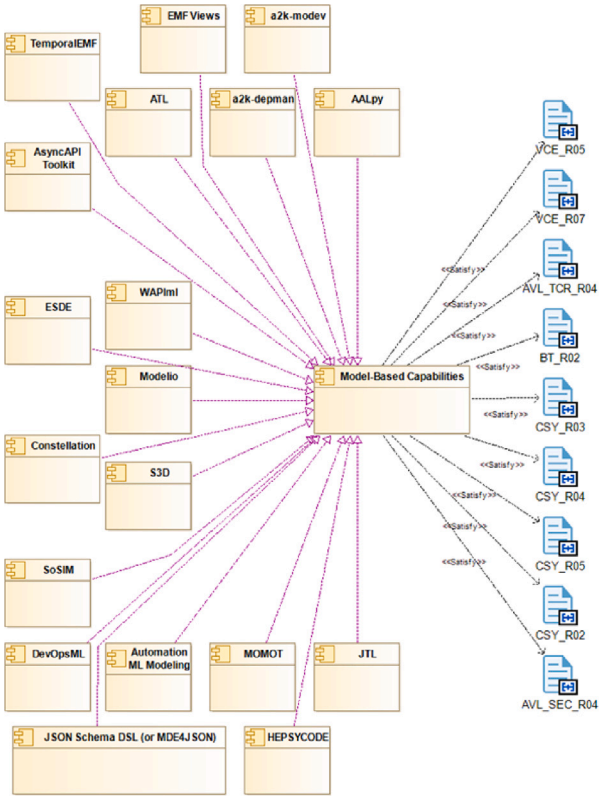

Fig. 8. Model-Based Capabilities component from the Core Tool Set, corresponding solutions and potential links with AIDOaRt case study scenarios.

the aggregation and abstraction of the particular case study data representations, using a standard UML notation. It has been built through collaboration across case study partners and modeling experts. Case study partners provided their corresponding case study data models, whereas a modeling expert executed the abstraction that resulted in the generic layer of the mega-model, with the support of stakeholders from all consortium members.

To illustrate the final mega-model artifact, Fig. 11 shows an excerpt of the *Testing data model* package. Abstract classes are the result of the bottom-up generalization of concepts from the case study data models (cf. Section 3.1); namely: *TestParameter*, *TestExecution*, *EvaluationCriteria*, *EvaluationMetric* and *TestResult*. They have links with the case study

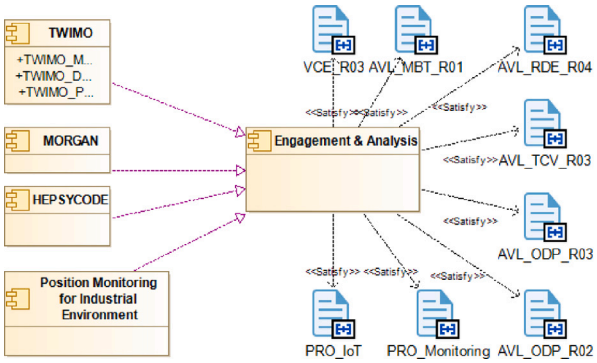

Fig. 9. Engagement & Analysis Capabilities component from the AI-Augmented Tool Set, corresponding solutions and potential links with AIDOaRt case study scenarios.

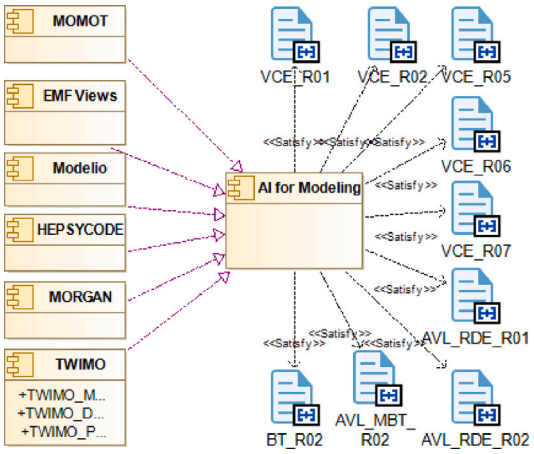

Fig. 10. AI for Modeling Capabilities component from the AI-Augmented Tool Set, corresponding solutions and potential links with AIDOaRt case study scenarios.

data entities they abstract, thus providing top-down and cross-domain traceabilities.

For instance, *TestResult* is the abstraction of the concrete classes *TestData*, *Report*, *Output* and *TestOutcome*, present in different case study domains. It represents the generated testing output data which would be used for further analysis. A set of *EvaluationMetrics* is defined to aggregate and measure the *TestResults* from a *TestExecution*. The actual values of the metrics are put in contrast with *EvaluationCriteria* to check if they conform to the expectations of the test case and are aligned to the business goals. In the scenario where a criterion is not met, it would be necessary to apply changes to the system or re-generate tests, to make sure that the system performance is still aligned to business needs.

EvaluationMetrics are not only used in test environments, but also in production, to continuously monitor the performance of the system, based on actual collected data. This is an example of relationship between packages of the AIDOaRt mega-model: the *EvaluationMetric* from the *Monitoring data model* is associated to *EvaluationCriteria* and *TestResult* from the *Testing data model*.

6. Evaluation

The objective of this section is to evaluate the proposed architecture considering the challenges discussed in Section 2.2. To this end, we demonstrate the application of the architecture in the context of the AIDOaRt project, considering case study scenarios and their development. In the rest of this section, we introduce the context of the experiment, the considered case studies, and the evaluation results.

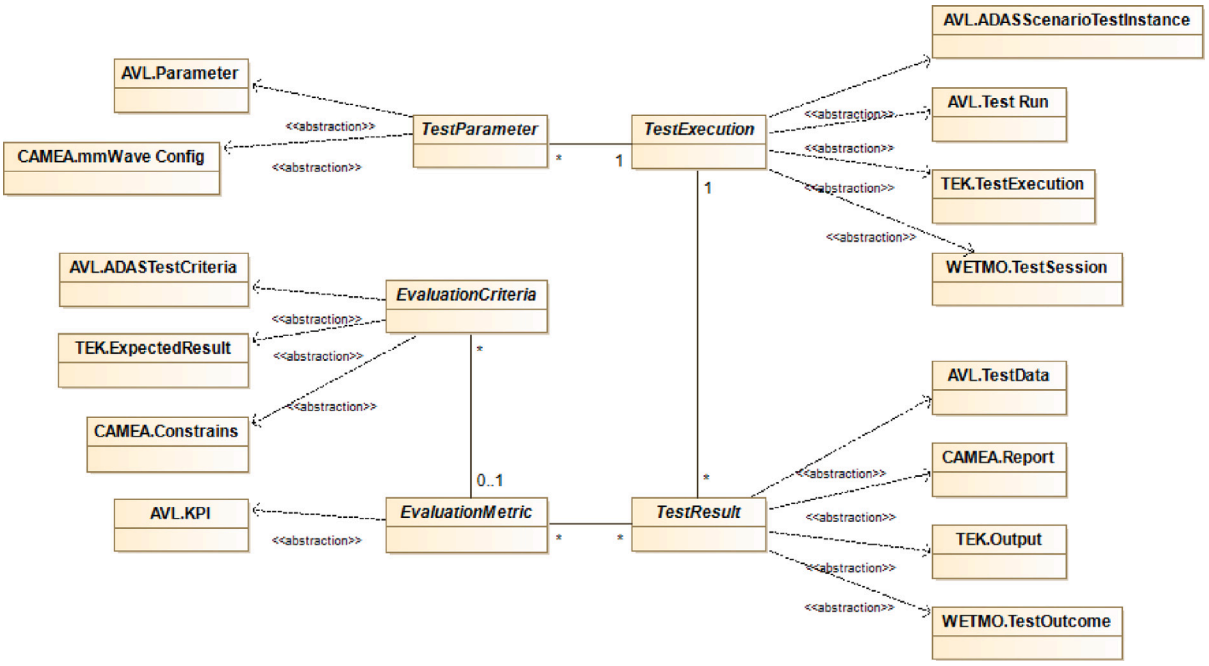

Fig. 11. An excerpt of the Testing data model, its abstract elements, and the links to the case study related concrete elements.

6.1. Context

The AIDOaRt project brings together a balanced consortium of 32 European partners (7 large companies, 12 SMEs, and 13 academic and R&D organizations). These partners are geographically covering 7 different European countries (Austria - AT, Czech Republic - CZ, Finland - FI, France - FR, Italy - IT, Spain - ES, and Sweden - SE). Among them, 10 industrial partners play the role of case study providers and end-users (the considered case studies and their technological domains are summarized in Table 1). Moreover, 9 industrial partners play the role of technology and service providers, while 13 partners drive the research activities (among the solution providers, three partners also played the role of case study providers).[4]

Case study providers proposed to tackle several case study scenarios related to cyber–physical systems in different application domains (e.g., railways, automotive, construction equipment, software, and communication systems). To realize these scenarios, the case study providers defined a quite varied and long list of 128 functional and data requirements. The requirements largely differed in terms of abstraction level, broadness, and coverage. Thus, a significant effort was needed to analyze and summarize these requirements in dedicated work groups led by partner experts in their respective MDE, AI/ML, and DevOps domains. As a result, we produced a short list of 30 generic high-level requirements classified w.r.t. the five key SE/DevOps activities we target: Requirements Engineering, Modeling, Coding, Testing, and Monitoring.

Solution providers proposed more than 50 research and commercial solutions as candidate tools, methods, and approaches to be part of the framework of AIDOaRt and to address the case study requirements. These solutions also vary in the services they offer (113), the consumed or produced input/output data, and their technical constraints in terms of data requirements, supported platforms, or deployment procedures.

When working on the evaluation of our proposed architecture, we decided to proceed in two complementary ways. Firstly, we worked on a qualitative evaluation based on the practical application of our architecture on several industrial case studies. This intends to demonstrate the applicability of our architecture, and is presented in Section 6.2. Secondly, we worked on a more quantitative evaluation based on mapping links between case studies, our architecture, and corresponding technical solutions. This intends to show how we addressed the targeted challenges (cf. Section 2.2), and is presented in Section 6.3.

6.2. Selected case studies

The case studies developed during the project (see Table 1) are considered for the overall evaluation. Moreover, we selected 3 of them as representatives to show specific details; for each of them, we extracted and reported part of the scenarios, and requirements, and developed solutions The integration of the AIDOaRt solutions to support the case studies, along with the exploration of potential collaborations between case study and solution providers, followed the *Integration mediation patterns* as described in Section 4. In each case study, specific requirements were abstracted using the *Generic Requirements Mediation Pattern* to create higher-level requirements independent of individual case studies. The data representations required to support the case study were also generalized and integrated into a mega-model following the *Data Engineering mediation pattern*. Finally, the requirements and potential solutions were mapped to the AIDOaRt components by applying the *AIDOaRt components mediation pattern*.[5]

6.2.1. AVL (AVL)

AVL is a leading mobility technology company for development, simulation, and testing in the automotive industry, and in other sectors. The company provides solutions for Real Driving Emissions (RDE). The current worldwide regulation for car homologation requires strict limits for emissions, battery range, and battery lifetime, prescribing tests with real driving conditions. To estimate the RDE of a vehicle driving along an arbitrary route under realistic driving conditions, the driver model needs to reproduce the human driver behavior as accurately as possible.

[4] Further details about the considered case studies and their technological domains, as well as solution providers and their contributions, have been published (Bruneliere et al., 2022).

[5] The interested reader can refer to the project public deliverables https://www.aidoart.eu/aidoart/results/deliverables.

Table 1
Case studies from AIDOaRt partners.

ID	Domain	Partner	Co.	Description
ABI	Automotive	Abinsula SRL	IT	Safety-Critical Systems in the Automotive Domain using Disruption Technology
AVL	Automotive	AVL List GmbH	AT	AI-supported Digital Twin Synthesis Supporting Secure Vehicle Development and Testing for Novel Propulsion Systems
BT	Railway	Alstom Transport AB	SE	DevOps for Railway Propulsion System Design
CAM	Transport and Smart mobility	CAMEA spol. s.r.o.	CZ	AI for Traffic Monitoring Systems
CSY	Railway	CLEARSY SAS	FR	Machine learning in interactive proving
HIB	Food services	HI Iberia Ingeniería y Proyectos S.L.	ES	AI DevOps in the Restaurants Business
PRO	Maritime	Prodevelop SL	ES	Smart Port Platform Monitoring
TEK	Electronic	Tekne SRL	IT	Agile process and Electric/Electronic Architecture of a Vehicle for Professional Applications
VCE	Automotive	Volvo Construction Equipment AB	SE	Data Modeling to Support Product Development Cost and Efficiency
WMO	Telecommunication	Westermo Network Technologies AB	SE	Automated Continuous Decision Making in Testing of Robust and Industrial-grade Network Equipment

This accuracy is critical for getting comparable vehicle behaviors either with a human or simulated driver. The current model, used by the company, is a simple rule-based parametric model, whose accuracy is not fully sufficient and needs to be improved.

DevOps Challenge: The company aims to improve the DevOps pipeline and extend the current testing and verification capabilities by exploring data-driven (AI-based) technologies, especially in the area of hardware testing and its security as well as autonomous driving technologies.

Scenario: Among the AVL scenarios addressed in AIDOaRt, we refer to two of them, namely "**AVL_RDE_UCS1**" and "**AVL_RDE_UCS2**" in Fig. 12. They concern the generation of a data-driven model of the driver's behavior based on environmental events (i.e. traffic signs, traffic signalization, traffic conditions, etc.) and applicable on any arbitrary test route. The case includes the generation of a driver profile that isolates vehicle-specific dynamics of the speed so that the driver behavior model is independent of the vehicle.

Requirements: As shown in Fig. 12, AVL formulated several requirements for the RDE case. The main requirements (identified as "**AVL_RDE_R01**" and "**AVL_RDE_R02**") concern, respectively, the "development of an ML model that, based on real driving recordings, is trained to simulate human-like driving given a target route, vehicle, and traffic conditions", and the "development of an AI method for providing better statistics of the environment based on the statistics of the real driving recording and data from digital map service". They relate the generic requirement "**GR Mod 4**" for the "use of semi-automatic model synthesis for design- and run-time verification". Moreover, the case requires "automate multi-source data analysis of the real driving test data such that the relevant features of the driver behavior can be clustered (e.g. highway driving, low-speed driving, cornering, braking, acceleration, ...)" ("**AVL_RDE_R03**"), that relates the "**GR Mon 5**" for the "monitoring tool using human-defined parameters". Finally, the case refers to the "**AVL_RDE_R05**" that requires the analysis of basic driver attributes, including acceleration/deceleration histogram, braking behavior, cornering behavior, and max speed preference. This relates the "**GR Mod 06**" for the "use of AI-based methods for easy configuration".

Development: Modeling the velocity profile of a human driver is not a trivial task. The very fact that the task is not discriminative due to hidden variables (e.g. traffic conditions, exact weather conditions, traffic signalization, ...) makes this generative ML problem challenging. Fig. 12 depicts an excerpt of the *Data Model*, as an instance of the data mega-model described in Section 5; it describes features and real-world measurements involved in the case study and the relationship between them.

Within the project, two different AI-augmented solutions, and correspondent *tools*, were implemented and applied to the case study, as follows

- **TWIMO** (AIDOaRt Consortium, 2022b) provides a conceptual framework to define domain-specific notations, used for the definition of human driver behavior and ML-based services. The tool provides also ML prediction methods; a Random Forest classifier was applied for the analysis and prediction of RDE in virtual environments.
- **AALpy** (Muškardin et al., 2021) provides a library implementing different active automata learning algorithms that support the learning of finite state models of black-box systems. A passive automata learning technique was applied to the measurement data collected from test drives by AVL to infer models in the form of Markov Decision Processes (MDPs).

The solutions developed for this scenario link several *Architecture Component* (cf. Fig. 13), e.g., *Engagement & Analysis* (data in the form of driving cycles is analyzed and transformed into a predictive behavioral model of human driving behavior), *AI for modeling* (suitable abstraction of the recorded data are defined, the method can be applied to other data sets to create models of different driver styles, *AI for testing*(AI-based solutions for Real Driving Emissions testing are developed).

Solutions Integration and Obtained Results: The tools described above, namely TWIMO and AALpy, are complementary and have been applied in the context of a demonstrator at AVL.

AALpy provided a prototype for learning probabilistic behavioral models of human driver behavior from a set of recorded driving cycles (i.e. test drives) enriched by static environment data such as the current speed limit and curvature. TWIMO provided a model-based framework for the specification of human driver behavior models and ML-based services.

Although the solutions have been evaluated by considering only a limited dataset at this stage, AVL experts have evaluated the results of the developed method, and this visual inspection has shown the following: 1. The driving profile generated by the proposed method significantly better models the driver speed profile on the motorway compared to the method currently in use in the company. 2. The

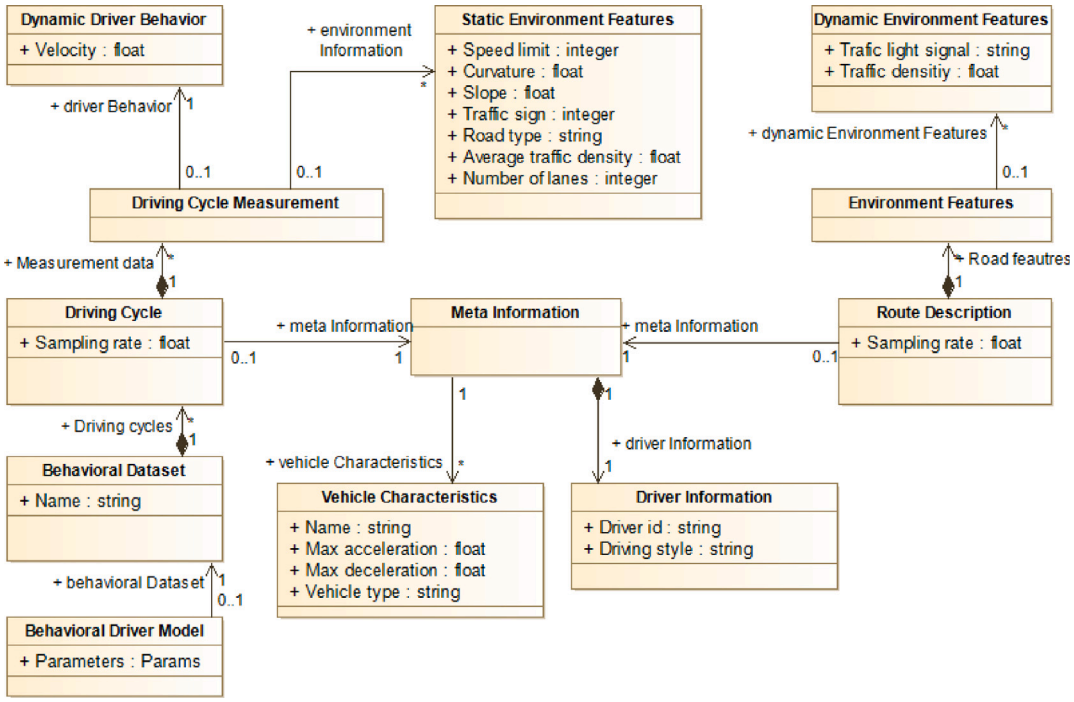

Fig. 12. An excerpt of the AVL RDE data model.

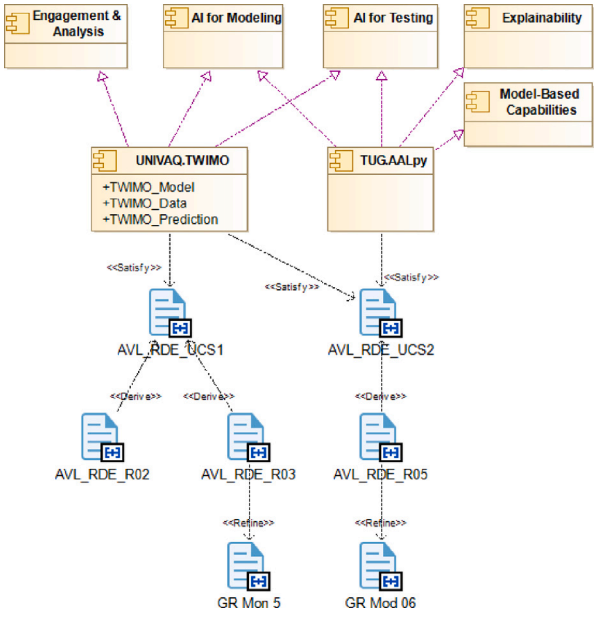

Fig. 13. AVL RDE case study scenarios, derived requirements, providing solutions, and instantiated architecture components.

general driving trend generated by the proposed method is hard to distinguish from the general driver trend of human driving. 3. Due to the approach's granularity, the proposed method's local speed profile does not show the variations seen in the speed profile of a human driver.

6.2.2. Alstom (BT)

Alstom (previously Bombardier Transportation, abbreviated BT) is a leading company in green and smart mobility worldwide, developing and marketing integrated systems that operate in rail transport markets.

DevOps Challenge: The company aims to automate data processing and transfer between different stages of the development process, multi-physics modeling, and test data correlation. The potential outcomes of such an endeavor are an improved design and development chain resulting in a more effective process, optimized solution customization, simulation, and test facilities, standards certification support, and reduction of the overall costs, i.e., time to market, maintenance, and life-cycle costs, monitored operations, energy efficiency, etc.

By using ML, the company aims to automate and improve its requirements engineering process using an ML solution that would analyze requirements, add adequate responses to each requirement, and automate the parametrization of specific models in the propulsion control system that depend on the physical properties of the corresponding hardware components. Today, this is done by manually iterating parameter settings against measured data during systems testing. While both case study scenarios have been addressed in AIDOaRt, we only present here the first case study scenario (labeled "**BT_UCS1**"), which tackles AI-augmented requirements engineering.

Scenario: Railway traction equipment consists of complex hardware and software elements that, in their aggregation, constitute complex cyber–physical systems. The business is driven by a bidding process, typically in the form of public procurement. Customers, i.e., railway rolling stock owners and/or operators, issue detailed specifications for the complete trains, some of which directly affect traction systems and others that can result in derived requirements. Despite the diversity between customers, most specifications address the same features and design aspects. However, there is a great diversity in how the requirements are formulated. Today, as a consequence, vast numbers of customer requirements are manually analyzed, allocated, and further broken down. This usually requires highly experienced bid and customer project engineers to be carried out effectively. The goal is to provide appropriate recommendations to the bid and project engineers in an automated manner based on datasets for actions and responses taken from previous projects. This includes finding requirement defects (such as ambiguities and vagueness), clustering or classifying the requirements (e.g., to allocate them to different teams or persons

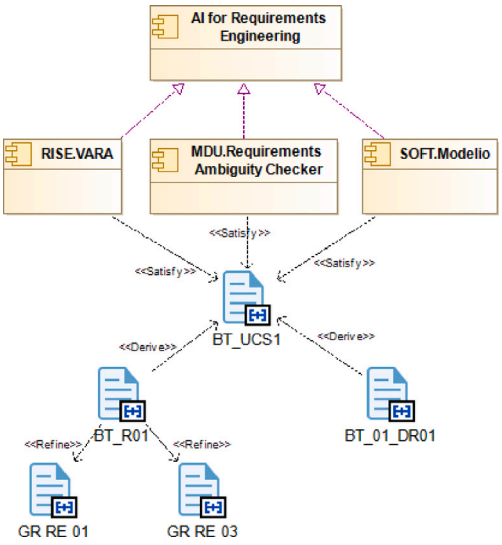

Fig. 14. Alstom (BT) case study scenario, derived requirements, provided solutions, and instantiated architecture components.

responsible), and responding to the requirements (can we comply with them or not?).

Requirements: As shown in Fig. 14, the main requirement (identified as "**BT_R01**") is formulated by Alstom as "providing NLP contextual analysis of requirements and matching against a database of responses/solutions" from previous projects. This requirement refines two generic requirements ("**GR RE 01**" and "**GR RE 03**"): the first aims to translate requirements of Cyber-Physical Systems from semi-structured language to formal language, while the second aims to analyze the CPS requirements expressed in formal language and to produce suggestions or prescriptions for the Requirements Engineer and the System Engineer (Bergelin and Strandberg, 2022).

In terms of data requirements (identified as "**BT_01_DR01**"), Alstom specified the need to safely import, store, and process requirements that are securely exported from tools used for requirements engineering (e.g., IBM Rational DOORS) as tabular datasets (e.g., CSV or Excel files). Consisting of design time static data, large datasets can be processed by smaller batches, and there is no particular need to handle big or real-time data.

Development: As shown in Fig. 14, Alstom's BT_UCS1 relies on solutions implementing the "AI for Requirements" component. These solutions are expected to analyze requirements, evaluate them, and recommend actions that are to be done by the requirements engineer. The component should be interoperable with third-party tools (e.g., IBM Rational DOORS) and provide data exchange capabilities. Alstom would use solutions implementing the "AI for Requirements" component to enhance the capabilities of requirements engineers and automate the requirements engineering process (cf. D4.1 and D4.2).

Among the solutions implementing the "AI for Requirements" component, Alstom has collaborated with MDU, RISE, and SOFT to benefit from their Requirements Ambiguity Checker (RAC), VARA, and Modelio's NLP4RE solutions (resp.):

- **RAC** (Bajceta et al., 2022) identifies ambiguous requirements from textual documents using a set of ambiguous keywords and patterns and a plethora of NLP and AI/ML techniques.
- **VARA** (Bashir et al., 2023) provides automated similarity analysis and feature reuse recommendations using NLP. It allows engineers to perform an automatic analysis of textual requirements for a new project and identify components and artifacts that can be reused from a previous project for the implementation of the new requirements based on similarity analysis.

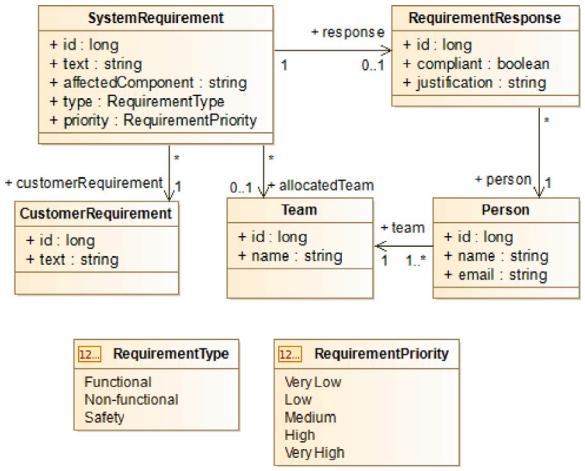

Fig. 15. Alstom (BT) case study data model.

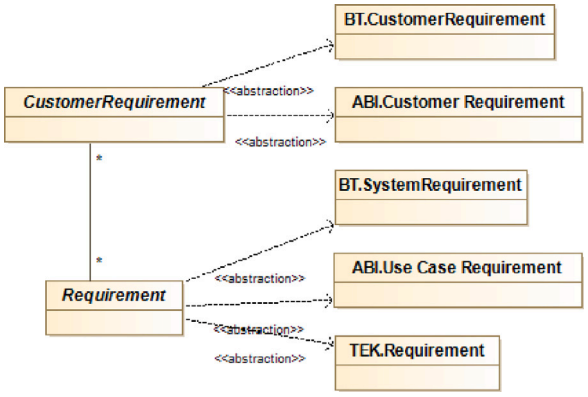

Fig. 16. Mega-model excerpt for Requirements Engineering.

- **Modelio's NLP4RE Module** (Nigmatullin et al., 2023) allows for automated requirements identification, extraction, and classification from textual documents with NLP and AI/ML techniques.

The BT case study data model is illustrated in Fig. 15. It supports the processing of system requirements written in natural language. Each system requirement specifies the system component affected, which customer requirement it addresses, and the recommended team allocation. Requirements are further elaborated with a requirement response.

This case study data model is mapped to the AIDOaRt mega-model (cf. Section 5.2), particularly to the *Requirements Engineering* package. Along with other specific data models, it served to abstract their common elements into generic ones as part of the AIDOaRt Data Representation component. An excerpt of the affected subset of elements of the mega-model is included in Fig. 16.

Solutions Integration and Obtained Results: These three solutions are complementary and can be integrated into a complex and comprehensive requirements engineering workflow. Indeed, after the automated identification and extraction of requirements from large textual documents, it is desirable to check for their ambiguity and propose to the requirements engineers ways of correcting, reviewing, or clarifying requests to the customers. Once the requirements are neatly clarified, automated similarity checks, classification, and allocation come in handy to provide the requirements engineers with valuable insights.

Currently, the **RAC** solution identifies ambiguous requirements from textual documents using a set of ambiguous keywords and patterns and various NLP & AI/ML techniques with an accuracy of 80%. **VARA**

enables automatic analysis of textual requirements for a new project and identifies components and artifacts that can be reused from a previous project to implement new requirements based on similarity analysis with an accuracy of 71%. The **NLP4RE** prototype identifies similar requirements with an accuracy of 60%. These results were thoroughly documented in AIDOaRt's deliverable D5.7 (AIDOaRt Consortium, 2023b).

According to Alstom (BT), the requirement **BT_R01** is fully covered by the proposed solutions. The success rate of requirement analysis (i.e., identification and ambiguity checking) and matching (i.e., similarity checking) will be determined during the validation of prototypes by the end of the AIDOaRt project and reported in its final deliverables. The solutions can be further extended beyond the current case study environment in other areas of requirements engineering within Alstom (BT), e.g., for the verification/validation phases, and in other domains such as software development and off-cycle R&D programs.

6.2.3. Volvo Construction Equipment (VCE)

VCE is a global manufacturing company in the construction machine industry. The company mainly works in the context of Product Line Engineering (PLE): much of the development effort regards managing and configuring the product lines of different machines for various missions and contexts. Indeed, most construction machines contain a large set of customizable or variable options in the system specification and across different sub-systems. Traditionally, managing the system architectures is primarily based on informal artifacts and documents. Such a practice can be quite inefficient given the modern system complexity. Thus, there is a need to enhance current methods and engineering practices at VCE in order to tackle the growth in system complexity.

DevOps Challenge: Model-based (System) Engineering or MBSE is a key paradigm to address this growing complexity while adding more advanced capabilities to continuous engineering workflows. However, previous experiences (Suryadevara and Tiwari, 2018) have shown that adopting MBSE and related processes is not straightforward in an industrial context. For example, there is a need for more maturity regarding the provided architectures, corresponding tooling capabilities, related advanced analysis capabilities, and activities focusing on continuous verification and validation (V&V). Still, the integration of AI and DevOps techniques is believed to further strengthen the use of models instead of informal artifacts and documents. The main global challenges are related to high-level analysis, modeling patterns, automated continuous workflows, and early validation processes.

Scenario: VCE is on a transformation journey focusing on the electrification of their construction machines, including both battery-electric and fuel-cell technologies. Maintaining quality is of utmost importance even during the transition period with fast prototyping and short lead times. This requires the application of new technologies not only in the final product but also during the design and development phases. In the AIDOaRt project, VCE proposed a set of case studies and challenges regarding the design and development of its future construction machines. The case studies notably include artifacts and related workflows currently followed. The main objective is to improve the current methods and practices via the combination of model-based methods with AI (and also DevOps in the next step).

More concretely, VCE provides a real case study based on the different VCE product lines[6] offering many types of potential configuration variants to customers. Currently, the AIDOaRt architecture has been applied by targeting the early phases of the system engineering life cycle at VCE. Indeed, there is generally a lack of harmonization of artifacts at this stage, and crucial information is often managed via documents and other non-formal descriptive artifacts. Engineering activities, like requirements specification and system-level detailed design and implementation, need more cohesive linking among the produced engineering artifacts. In this regard, there is a need to adopt robust modeling techniques and practices, i.e., opting for prescriptive rather than descriptive approaches. In this way, it would be possible to boost the automation of the engineering process by connecting artifacts of different stages, to propagate consistently and share engineering decisions across machine-readable artifacts to enable a more scalable and interoperable workflow.

Requirements: As shown in Fig. 17, the main requirement (identified as "**VCE_UCS_01**") for the selected VCE scenario has been derived as two sub-requirements ("**VCE_R05**" and "**VCE_R07**"). These two derived requirements actually refine two generic requirements related to Modeling capabilities (respectively "**GR_Mod_09**" and "**GR_Mod_07**"). In the first case, the goal from the VCE perspective is to customize standards modeling languages (e.g. SysML, AutomationML) in order to develop proper VCE system, software, and data architecture models. In the second case, the aim is to (1) provide AI-based recommendation support to the VCE engineers in order to help them improve their systems models, and (2) integrate this recommendation support for continuous model configuration and delivery to improve variability management; In addition to what is shown in Fig. 17, the selected VCE scenario also includes the preservation of the requirements coming from the VCE original artifacts and thus semi-automatically extracted from them.

Development: As shown in Fig. 17, the currently proposed solution integrates several model-based tools from the AIDOaRt Data Engineering Tool Set, AIDOaRt Core Tool Set and AIDOaRt AI-Augmented Tool Set (some tools belonging to multiple tool sets due to their different capabilities). Among them, we can notably mention:

- **AutomationML Modeling** (Mayerhofer et al., 2018) providing an implementation of the AutomationML standard modeling language via the CAEX workbench;
- **Modeling Process Mining Tool/MER** (Dehghani et al., 2020) offering the support for capturing events in the context of graphical model editing;
- **EMF Views** (Bruneliere et al., 2015) allowing to federate the different involved models (SysML, AutomationML, traceability) as model views;
- **MORGAN** (Di Sipio et al., 2023) providing the needed recommendation support over the models of interests (cf. the previous item).

Moreover, whenever needed, the solution also integrates the use of open-source tools, such as Papyrus for design modeling, as well as a few components developed from scratch (e.g., specific model-to-model transformations).

Fig. 18 depicts a partial view of the VCE Data Model that focuses on the model recommendation aspects, as directly directed to the presented scenario. The complete VCE Data Model can be found from a specific AIDOaRt deliverable (AIDOaRt Consortium, 2023c). It describes all the features and real-world measurements involved in the VCE case study in general, as well as the relationships between them.

As mentioned before, the proposed integrated solution developed for this scenario heavily relies on the **Model-Based Capabilities** offered by the Data and Modeling component from the AIDOaRt Core Tool Set. Moreover, this solution also directly links to the several components from the Data Engineering Tool Set and AI-Augmented Tool Set. Notably, **AI for Modeling**, **Engagement & Analysis**, **Ingestion & Handling** relate to AI for assisting modeling activities (cf. the used recommendation support), for generating instance models from higher-level architectural ones, and for synchronizing different views based on their contributing models.

[6] https://www.volvoce.com/europe/en/products/articulated-haulers/a60h/.

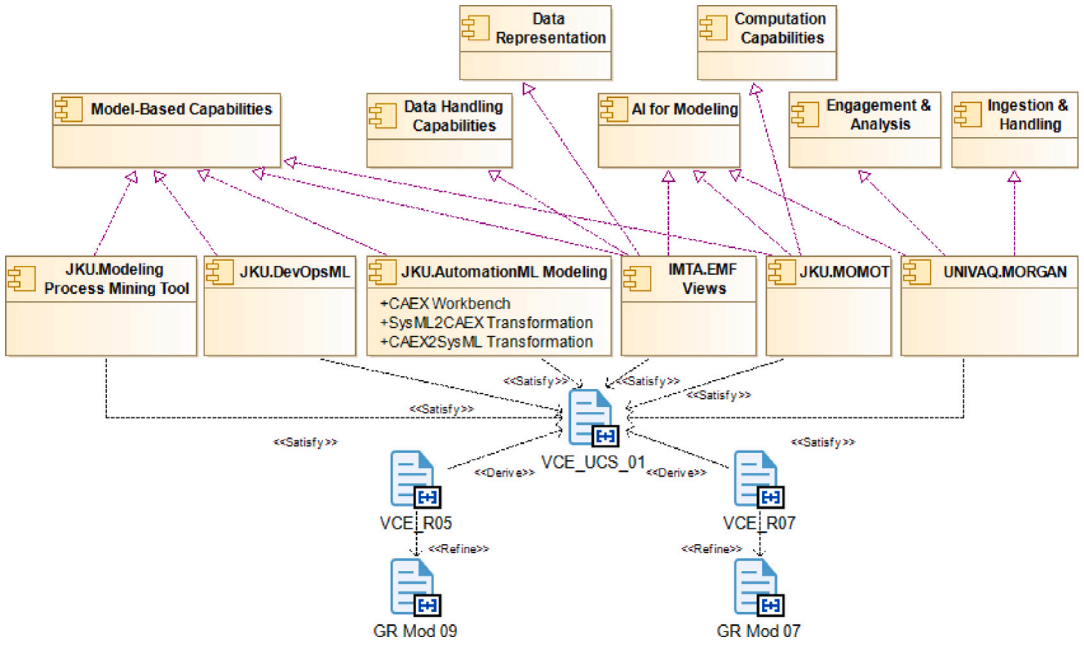

Fig. 17. VCE case study scenario, derived requirements, provided solutions, and instantiated architecture components.

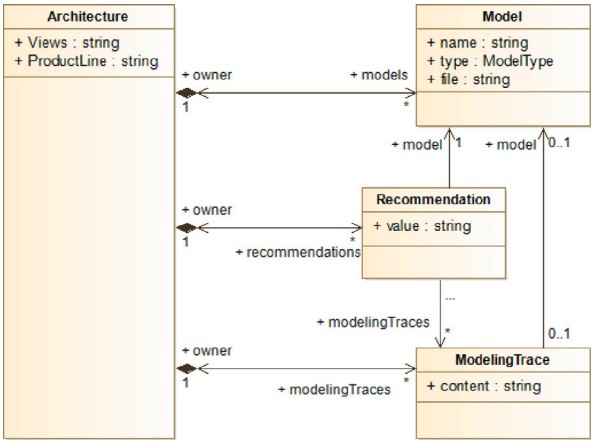

Fig. 18. An excerpt of the VCE data model (with a focus on the model recommendation aspects).

Overall, at VCE, the development of this case study is contributing to the automation of several important modeling activities, an increase in the system design and development velocity via the use of such model-based techniques, and ultimately the further re-use of the same architectural models in the context of different projects.

Solutions Integration and Obtained Results: The four tools mentioned before, namely AutomationML Modeling, MER, EMF Views, and MORGAN, have been chained together in practice and applied in the context of a demonstrator at VCE. Concretely, it allowed to rely on existing standard modeling languages (UML, SysML, AutomationML) while customizing the notations to enable VCE-specific concepts (cf. "**VCE_R05**"). Furthermore, it also facilitated the re-use of, and learning from, legacy artefacts created by the VCE engineers within past projects (cf. "**VCE_R07**").

The modeling recommendations provided by the combination of MER and MORGAN have been evaluated by conducting an initial user study. In particular, we involved four different types of engineers from VCE (Verification Engineer, Software Engineer, System Architect, and System Engineer). Each session roughly took one and a half hour and we collected their feedback in a structured questionnaire. Overall, the participants are satisfied with the examined aspects. A deeper evaluation involving more participants is planned to be performed in the near future.

The continuous modeling support provided by EMF Views has been evaluated by considering only a limited set of models at this stage. In particular, we have been able to experiment with the specification and building of an initial "complete" view interconnecting SysML, AutomationML models (as design models) and more recently a FMU model (as a runtime model). This already showed promising results in terms of model federation capabilities within the VCE context.

6.3. Evaluation results

In this section, we evaluate the proposed AIDOaRt architecture by considering the targeted challenges (cf. Section 2.2). We notably look at how the involved industrial case studies, and the research and commercial solutions, are actually integrated within the framework of the proposed architecture, according to architectural and technical demands.

6.3.1. Architectural challenges

To evaluate the architectural challenges, we demonstrate that the components and functionalities of the proposed architecture are able to (1) cover case study requirements, and (2) be implemented by corresponding technological solutions providing the needed capabilities.

A1. Heterogeneity. Fig. 19 gives an overview of the mapping from the case study requirements to the AIDOaRt architecture's components, and their functionalities (as described in Section 3). The provided chart shows, per case study provider, the number of functional and data requirements that can be possibly satisfied by each architecture component, i.e., by one or more of its functional interfaces. This mapping has been specified directly by the case study providers, based on their deep knowledge and expertise in their respective case studies.

Going into more detail, the case studies presented in Section 6.2 show how heterogeneous requirements, in different domains and for different goals, have been satisfied by the architecture. For example, AVL and VCE have very different requirements and require

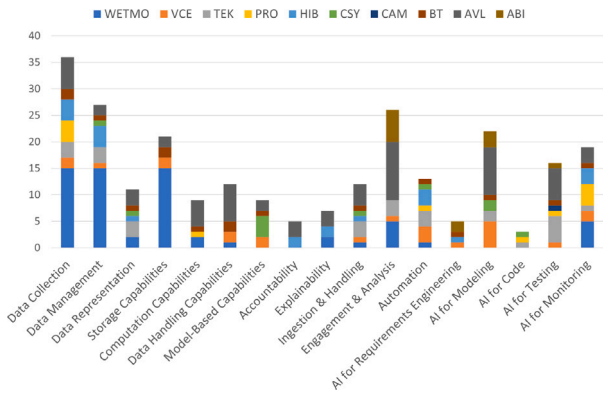

Fig. 19. Mapping of case study requirements to the architectural components. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

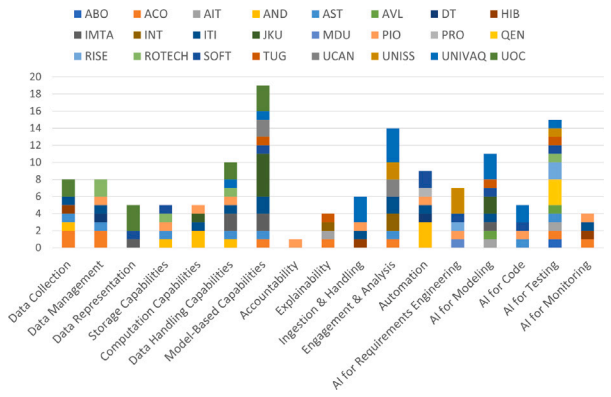

Fig. 20. Mapping of solutions to the architectural components. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

different functional capabilities: the first focuses on engineering a driving behavior modeling and prediction service, and the other requires the use of model-driven and data-driven methods in the company's engineering workflows in general.

The high connectivity of the architecture's components to both the case study requirements and solutions shows the global alignment of the architecture with the general scope and objectives of the AIDOaRt project. Moreover, it also somehow attests to the overall relevance of the current architectural design and specification. A large number of requirements and solutions, and their wide spectrum of domains and topics, can be fully covered with the relatively small number of generic architecture components and their functional interfaces. Furthermore, we can assess that the architecture could be reused and extended in the context of other organizations in various domains.

A2. Flexibility. Fig. 20 gives an overview of the mapping from the solutions to the AIDOaRt architecture's components, and their functionalities (as described in Section 3). The provided chart shows, per solution provider, the number of solutions that can possibly implement each architecture component, i.e., one or more of its functional interfaces. Within the AIDOaRt project, we considered a total number of 54 technological solutions provided by 22 partners.

This mapping has been specified directly by the solution providers, based on their knowledge of their respective solutions. For example, among the selected case studies, the AVL scenario (Fig. 13) has been supported by more than one solution implementing the same capabilities. As a result, we can see that the proposed architecture is able to be realized via the different solutions made available by the various AIDOaRt project partners. This architecture can be deployed in the context of AIDOaRt of course, but also possibly in the context of other projects in the future.

A3. Integration. Fig. 21 summarizes the previous results and indicates both the number of solutions implementing one or more of the functional interfaces of the architecture components and the number of case study's functional and data requirements that are possibly satisfied by each architecture component. This shows how, as a result of applying the *AIDOAaRt components mediation pattern*, a large number of requirements and solutions capabilities can be organized together and abstracted by a relatively small number of high-level components and functional interfaces. Furthermore, the generalization of requirements and data representations, through the *Generic Requirements Mediation Pattern* and the *Data Engineering mediation pattern* respectively, further contributes to the overall integration strategy. By employing the *Generic Requirements Mediation Pattern*, 126 case study-specific requirements were organized and mapped into 47 generic requirements. At the same time, the *Data Engineering Mediation Pattern* facilitated a standardized approach for data representation, easing the integration of disparate data sources.

A4. Collaboration. The goal of this other part of the evaluation is to show in practice that the components and functional interfaces of the proposed architecture can cover potential industrial requirements and solutions capabilities. Notably, we want to show the high connectivity of the AIDOaRt architecture by identifying potential collaborations between case study providers and solution providers.

The heat maps of Fig. 22 display both the current collaboration links between solution providers and case study providers (left) and their potential implicit/indirect links (right). We identified these links by analyzing the mapping of the solutions and requirements to the architecture components and interfaces. The numbers indicate the number of case study scenarios being tackled by a given solution. The higher the number, the darker the cell's highlight color (zeros are omitted for the sake of simplicity). These heat maps show that 91% of the actual collaborations were already identified by the indirect links as potential relations. However, the opposite is not true. Indeed, AIDOaRt partners are in the process of studying the list of potential links to either confirm or refute them. Confirmed links would lead to effective collaborations, whereas refuted links would be provided with a justification for not reaching common agreements. For instance, a potential collaboration may be discarded due to the divergence between the solution and the requirements on some technical constraints. This can also be due to the limited resources of a solution or case study provider.

A further analysis of these direct and indirect links between solutions and case studies can be made to provide some recommendations. For example, we are currently exploiting these mappings to identify new potential collaborations for the next milestones of the project.

6.3.2. Technical challenges

To evaluate the technical challenges, we demonstrate that the components and functionalities of the proposed architecture can (1) cover the technical demands identified, and (2) be implemented by corresponding technological solutions providing the needed capabilities.

T1. Data management. Several architectural components and capabilities have been defined to manage, store, and analyze large amounts of data. In particular, the dedicated *Data Engineering*

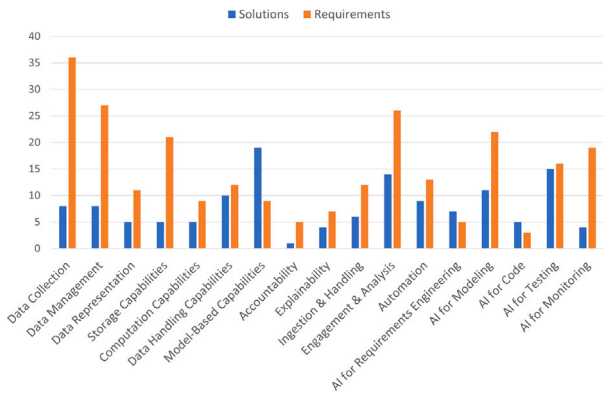

Fig. 21. Summary of the solutions and requirements mappings to the AIDOaRt architecture components. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Data Engineering Tool Set, number of provided solutions and mapped requirements.

Components	# Provided solutions	# Mapped requirements
Data collection	11	36
Data management	10	27
Data representation	10	11

Table 3
Core Tool Set, number of provided solutions and mapped requirements.

Components	# Provided solutions	# Mapped requirements
Storage capabilities	3	21
Computation Capab.	9	9
Data Handling Capab.	10	12
Model-based Capab.	19	9
Accountability	1	5
Explainability	5	7

Tool Set component offers functional capabilities for the collection, management, and representation of data. To illustrate this, Table 2 shows the number of implemented solutions/tools and the number of mapped requirements. Moreover, the data management capabilities are related to the *Data Handling Capabilities* of the *Core Tool Set* (cf. Table 3), as well as the *Ingestion & Handling* component of the *AI-augmented Tool Set* (cf. Table 4). Employing specific interfaces, these independent entities (and their implementation) can interact and communicate with each other. For instance, Fig. 17 illustrates how some of the developed solutions implemented the *Data Handling Capability*, *Data Representation* and *Ingestion & Handling* components.

T2. Modeling support.
Within the *Core Tool Set* component, several solutions/tools have been implemented for efficient handling of the various kinds of data artifacts (mostly data models), obtained from the *Data Engineering Tool Set*. A key goal of the *Core Tool Set* is also to provide support for all the other kinds of available software and system engineering models. Table 3 shows the number of implemented solutions/tools and the number of mapped requirements. Employing specific interfaces, these independent entities (and their implementation) can interact and communicate with each other, for instance, interconnected models can then be used to feed the AI-augmented features provided in the *AI-augmented Tool Set* component.
Among the others, the VCE scenario previously described in Section 6.2 is a fitting example of how the *Model Based Capability* and *AI for Modeling* components have been combined to develop solutions and meet the scenario's requirements (see Fig. 17).

Table 4
AI-augmented Tool Set, number of provided solutions and mapped requirements.

Components	# Provided solutions	# Mapped requirements
Ingestion & Handling	7	12
Engagement & Analysis	16	39
Automation	13	13
AI for Requirements	7	5
AI for Monitoring	4	19
AI for Modeling	13	24
AI for Coding	6	3
AI for Testing	17	16

T3. AI-powered DevOps. A large set of AI-based solutions have been implemented in the project. In particular, Table 4 shows the number of implemented solutions/tools and mapped requirements of the *AI-augmented Tool Set* component. As depicted in Fig. 2, this component combines sub-components and related capabilities/interfaces, that can interact and communicate to develop specific solutions. For instance, the AVL scenarios (cf. Fig. 13) show how the *AI for Modeling*, *AI for Testing* and *Engagement & Analysis* components have been implemented. Another example is the BT scenario (cf. Fig. 14) where the *AI for Requirement Engineering* component has been implemented.

7. Threats to validity

In this section, we discuss the threats to validity related the current evaluation of our proposed architecture. We consider the following dimensions: construct, internal, external, and conclusion validity threats. We also discuss the actions we already took in order to mitigate them.

7.1. Construct validity

Construct validity relates to the accuracy of a measurement or a test assessing the targeted attribute or concept. In our current evaluation, the selected attributes concern different challenges that have been identified from the literature. Given the diverse definitions of these attributes, we have explicitly defined them in the paper in order to clearly delimit their scope. Moreover, we have not included qualitative measurements such as questionnaires in the presented work because they have already been largely used in another context (Sadovykh et al., 2021, 2024). Instead, for each of the defined challenges, we have employed quantitative metrics to assess the extent to which AIDOaRt provides support for or addresses these specific challenges.

7.2. Internal validity

Internal validity threats are related to possible wrong conclusions about causal relationships and potential methodological errors. To mitigate this threat, a unique global architecture model was created, built, shared between all the project's partners and then maintained, evolved during the whole project's duration. More importantly, this architecture model is always considered as the singular source of truth. The construction and maintenance of this model involved the definition of micro-tasks, which were strictly monitored and assessed by each responsible partner (Sadovykh et al., 2021, 2024). This notably allows to guarantee a high level of consistency in the collected data, and to mitigate the potential for misinterpretations or erroneous inputs at the same time.

	ABI	AVL	BT	CAM	CSY	HIB	PRO	TEK	VCE	WEM
ABO		3	1	1						
ACO							3			
AIT		2								
AND						1				
AST								1		
AVL			1							
DT									2	
HIB						1				
IMTA									2	
INT	9									
ITI							2			
JKU									3	
MDU			1							
PIO										
PRO							1			
QEN										
RISE			1							
ROTECH								3		
SOFT			1			1			1	
TUG		2	1							
UCAN								2		
UNISS	5									
UNIVAQ		4						1	1	
UOC							1			

	ABI	AVL	BT	CAM	CSY	HIB	PRO	TEK	VCE	WEM
ABO	1	2	1				1	2	1	
ACO	4	15	6	1	7	8	7	6	4	8
AIT	3	10	2	1	2	2	1	2	3	3
AND		8	6		9	9	8	9	8	12
AST	3	11	3	1	6	4	5	2	3	4
AVL	3	11	1	1	2		1	2	3	
DT		1	1		4	4	1	3		4
HIB		6	3		3	3	5	3	1	4
IMTA	3	17	5		12	1		3	3	4
INT		16	2			2		6	2	11
ITI	6	29	6		16	4	5	6	8	11
JKU	6	36	8		32		1	4	12	3
MDU	2					1			1	
PIO	2	6	3		5	4	5	3	3	4
PRO			1		3	3		3	2	1
QEN	3	9	3	3			3	6	3	
RISE	4	6	2	2		1	2	4	3	
ROTECH	1	6	4	1	2	8	1	6	5	12
SOFT	3	13	4	1	9	6	3	6	5	6
TUG	3	10	1	1	6	2	1	2	3	3
UCAN	6	20	4		12			4	4	8
UNISS	6	17	6	1		3	1	8	6	8
UNIVAQ	12	51	9	1	17	3	3	11	9	16
UOC		16	5		18	9	9	9	6	10

Fig. 22. Collaborations and potential indirect links between solution and case study providers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.3. External validity

External validity threats are related to the ability to generalize the currently obtained results. The evaluation presented in this paper has been conducted on various different and real industrial case studies in order to mitigate threats related to the selected participants (e.g. non-representative subjects). Nevertheless, our evaluation has been conducted on companies that are partners of the AIDOaRt project and with which we have regular (and good) contacts. To mitigate this, the companies who participated in the evaluation are from diverse application and/or technological domains, in addition to being of various sizes. Thus, this ensures that the presented findings are not confined to a specific industry or organizational scale.

7.4. Conclusion validity

Conclusion validity is related to the reliability of the conclusions drawn from the results. To ensure reliable results, our evaluation is based on a significantly large set of 10 different industrial case studies. To address these case studies, 54 different technological solutions which have been designed, developed, (re)used and/or combined in varied ways. Nevertheless, we have been cautious to avoid making too strong statistical conclusions from the general observations we drawn as a result of our current evaluation. Instead, the findings we made are mostly used to demonstrate the feasibility, applicability and usefulness of the proposed approach.

8. Discussion

In addition to the experiment and corresponding evaluation reported in Section 6, we also extracted some general lessons learned from our overall experience of working on/with the proposed AIDOaRt architecture. We believe these findings could be possibly generalized to other kinds of collaborative projects targeting various kinds of systems in the context of different application domains. We hope these to be useful to the architecture and modeling communities, as well as more globally to the whole Software Engineering community at large.

A relevant support for project development. The feedback already collected at this advanced stage of the project (i.e., while reaching the last months of the project) shows that our architecture is generally beneficial from a project development perspective. Indeed, the identified components, their interfaces, and the corresponding cartography of supporting solutions appear to make easier the design and development of various technical environments suitable for different kinds of practical industrial scenarios. This has already been experienced with a satisfying level of success in the context of most of case studies from the AIDOaRt project, and this is currently being applied also within the latest remaining AIDOaRt case studies.

An interesting collaboration enabler. More globally, we have observed that relying on such a shared architecture fosters inter-partners collaboration, particularly between companies but also with academics (in all possible ways). In practice, the availability of the architecture and supporting solutions already facilitated the creation of new direct collaborations inside AIDOaRt (e.g., between a case study provider and solution providers relevant to his specific context). Moreover, it allowed companies to identify new challenges and potential solutions going beyond the scope of the current architecture and project. For example, a solution provider or an academic researcher can offer capabilities on other topics of interest for a given case study provider.

A virtuous iterative process. The elaboration of our architecture has been realized by following a combination of top-down and bottom-up approaches. On the one hand, based on the result of a first collaborative design effort carried out by a reduced number of partners, the AIDOaRt core team proposed an initial version of the architecture to the other partners. On the other hand, and in parallel, the case study providers started to work on the description of their case study requirements and scenarios. Then, all partners reviewed these elements and checked the alignment between the proposed architecture and the described case studies. Such early feedback, and the direct involvement of all the partners, allowed to iteratively enrich and improve both the architecture and the case study descriptions.

The required learning curve. From the beginning of the project, we noted that some AIDOaRt participants had previously limited experience in modeling and/or AI principles, concepts, and techniques. This resulted in difficulties for them to initially catch up with some of the characteristics of our proposed architecture. It also appears that the way we designed and maintained our architecture, by heavily relying on the Modelio tooling, was sometimes a source of misunderstanding in the first half of the AIDOaRt project. However, this has since been resolved by the AIDOaRt core team and the Modelio support team. In order to overcome these initial difficulties, we notably relied on multiple online presentations and hands-on sessions for the different project partners. Now that we are reaching the last months of the project, we can state that the use of such a model-based architecture has proven to be globally very positive for the different involved project's partners.

9. Related work

The AIDOaRt project was born as a direct follow-up of the successful MegaM@Rt2 project (Afzal et al., 2018). MegaM@Rt2 aimed at realizing a scalable model-based framework for continuous development and runtime validation. Thus, it addressed one of the key aspects of AIDOaRt, i.e., the continuous development (or DevOps) of complex industrial Cyber-Physical Systems (CPSs). However, AIDOaRt and its architecture intend to go much further. Indeed, the objective is to consolidate and improve DevOps practices for such CPSs via the additional use of AI/ML techniques.

In this context, the authors reported on their study of the state-of-the-art at the crossroads of the three areas of interest for AIDOaRt, namely MDE, AI/ML, and DevOps (Berardinelli et al., 2022). The results show a more robust integration between MDE and DevOps compared to other pairs, such as AI/ML and DevOps or MDE and AI/ML. Most MDE+DevOps studies presented model-based approaches explicitly applied in a DevOps context (e.g., generating a textual CI/CD pipeline script from a model artifact via model-to-text transformation) (Colantoni et al., 2020; Bordeleau et al., 2020a; Hugues et al., 2020). In contrast, studies that focus on AI/ML and DevOps predominantly focus on the application of AI/ML techniques to enhance DevOps artifacts (e.g., updating steps of a CI/CD pipeline based on outcomes of ML algorithms applied to runtime data of the delivered application) (Azizi, 2021; Beneventi et al., 2017). Lastly, we identified several studies presenting model-based approaches explicitly applied in AI/ML context (e.g., a metamodel/grammar specifying a domain-specific language for neural network specification) (Zhao et al., 2017; Jahić et al., 2023).

However, the most challenging question remains related to the adoption and integration of MDE, AI/ML, and DevOps principles and practices altogether. In fact, to the best of our knowledge, only a few approaches intend to actually realize such an integration in practice.

For example, in Castellanos et al. (2020) and Castellanos et al. (2021), the authors propose a Domain-Specific Model (DSM) and related DevOps practices to design, deploy, and monitor performance metrics in Big data analytics (BDA) applications. Such applications use ML algorithms to extract valuable insights from large, fast, and heterogeneous data sources. In this context, new challenges include ensuring sufficient performance levels of the data-driven algorithms even in the presence of large data volume, velocity, and variety (3Vs). The proposed approach also includes a design process and a framework to define architectural inputs, software components, and deployment strategies through integrated high-level abstractions.

In van den Heuvel and Tamburri (2020), the authors propose a framework that treats model abstractions of AI/ML models as first-class citizens and promotes transparent data detection, model verification, and model management. This framework is the first attempt to incorporate requirements stemming from intelligent enterprise applications into a logically structured architecture. The goal is to instrument applications with intelligence and continuously deploy, test, and monitor intelligent applications.

In Bersani et al. (2019), the authors propose a tool that retrieves data-intensive topologies for (1) anti-pattern analysis, thus allowing the detection of known and established design anti-patterns for data-intensive applications, and (2) transparent formal verification transposing the recovered data-intensive topology models into equivalent formal models to verify temporal properties, such as basic queue-safety clauses. This tool can be part of a DevOps pipeline dedicated to data-intensive solutions. It can be used for instrumenting the continuous refactoring of the data-intensive application by studying the application structure and the underlying topology to improve its operational characteristics.

In contrast with these quite specific works, we rather propose a general architecture supporting systems engineering and continuous delivery activities by combining AI-augmented, automated model-based engineering and DevOps. However, the investigated state-of-the-art (Berardinelli et al., 2022) provided the solid ground on which we based the proposed architecture and related applications on case studies.

10. Conclusion

This paper presented the proposed AIDOaRt architecture that provides the foundations of a model-based framework for developing AI-augmented solutions incorporating methods and tools for continuous software and system engineering and validation. Thus, the key characteristic of our architecture is that it allows leveraging the advantages of both AI and MDE approaches and techniques in a DevOps context. The paper illustrated how this architecture is implemented by different technical solutions within the context of several real industrial case studies coming from the AIDOaRt project. From this experimental evaluation and the corresponding collected data, we analyzed the capabilities of the proposed architecture regarding both architectural and technical challenges.

The paper reported on the applicability and usefulness of the proposed AIDOaRt architecture to tackle the integration of solutions in the context of the AIDOaRt case studies. Overall, we experienced in practice that our approach is beneficial from a project development perspective. The proposed architecture proved to be a key element for the success of the overall project outcomes, especially in terms of integration and satisfaction of requirements. It provided a complete picture of the requirements and required capabilities and a valuable integration strategy. Indeed, the identified components and interfaces and the corresponding cartography of the supporting solutions appear to make the iterative design and development of technical environments easier and more suitable for different kinds of real industrial scenarios. Some difficulties were encountered in the early stages of adopting and understanding the architecture, such as initial difficulties in understanding a model-based approach (which were overcome during development) or difficulties in implementing specific collaborations within the architecture. After an initial learning phase, the project's partners began to benefit fully from the approach, which also proved an interesting tool for driving collaborations.

Another limitation is the lack of an architecture quality analysis assessment. According to the ATAM approach (Kazman et al., 2000) and the recent survey in Silva et al. (2023), the evaluation of architecture quality must refer to an overall framework with quality metrics, attributes, and characteristics, or survey-based evaluations or interviews. This work is under development, and we plan to publish a future contribution when the analysis results are ready.

CRediT authorship contribution statement

Romina Eramo: Writing – review & editing, Writing – original draft, Supervision, Methodology, Data curation, Conceptualization. **Bilal Said:** Writing – review & editing, Resources, Methodology, Data curation, Conceptualization. **Marc Oriol:** Writing – review & editing, Methodology, Data curation. **Hugo Bruneliere:** Writing – review & editing, Methodology, Data curation, Conceptualization. **Sergio Morales:** Writing – review & editing, Methodology, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The work presented in this paper is funded by the ECSEL Joint Undertaking (JU)under grant agreement No. 101007350 (AIDOaRt project). The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, Spain.

References

Afzal, W., Bruneliere, H., Di Ruscio, D., Sadovykh, A., Mazzini, S., Cariou, E., Truscan, D., Cabot, J., Gómez, A., Gorroñogoitia, J., et al., 2018. The megam@rt2 ecsel project: Megamodelling at runtime–scalable model-based framework for continuous development and runtime validation of complex systems. Microprocess. Microsyst. 61, 86–95.

AIDOaRt Consortium, 2022a. D2.2 - Data Collection and Representation – Intermediate Version, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2022b. D4.1 AIDOaRt AI-Augmented Toolkit - Initial Version, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2022c. D2.1 - Data Collection and Representation - Initial version, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2022d. D5.5 Use Cases Requirements and Scenarios Evaluation Report, Deliverable. H2020-KDT AIDOaRt Project, URL https://www.aidoart.eu/aidoart/results/deliverables.

AIDOaRt Consortium, 2022e. D 3.2 - AIDOaRt Core Infrastructure and Framework - Initial Version, Deliverable Ref. Ares(2022)3313183-29/04/2022. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2022f. D 3.3 - AIDOaRt Core Infrastructure and Framework - Intermediate Version, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2023a. D4.2 AIDOaRt AI-Augmented Toolkit - Intermediate Version, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2023b. D5.7 Use Cases Evaluation Report 1, Deliverable. H2020-KDT AIDOaRt Project, URL https://www.aidoart.eu/aidoart/results/deliverables.

AIDOaRt Consortium, 2023c. D2.3 - Data Collection and Representation – Final Version, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2023d. D 3.4 - AIDOaRt Core Infrastructure and Framework - Final Version, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

AIDOaRt Consortium, 2024. D4.2 AIDOaRt AI-Augmented Toolkit - Final Version - to appear, Deliverable. H2020-KDT AIDOaRt Project, URL https://sites.mdu.se/aidoart/results/deliverables.

Albawi, S., Mohammed, T.A., Al-Zawi, S., 2017. Understanding of a convolutional neural network. In: 2017 International Conference on Engineering and Technology. ICET, pp. 1–6. http://dx.doi.org/10.1109/ICEngTechnol.2017.8308186.

Azizi, M., 2021. A tag-based recommender system for regression test case prioritization. In: 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops. ICSTW, pp. 146–157. http://dx.doi.org/10.1109/ICSTW52544.2021.00035.

Bajceta, A., Leon, M., Afzal, W., Lindberg, P., Bohlin, M., 2022. Using NLP tools to detect ambiguities in system requirements - A comparison study. In: NLP4RE 2022: 5th Workshop on Natural Language Processing for Requirements Engineering @ REFSQ (CEUR Workshop Proceedings). Vol. 3122, pp. 1–10.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F., 2020. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Inf. Fusion 58, 82–115.

Bashir, S., Abbas, M., Ferrari, A., Saadatmand, M., Lindberg, P., 2023. Requirements classification for smart allocation: A case study in the railway industry. In: 2023 IEEE 31st International Requirements Engineering Conference. RE, IEEE, pp. 201–211.

Beneventi, F., Bartolini, A., Cavazzoni, C., Benini, L., 2017. Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017. pp. 1038–1043. http://dx.doi.org/10.23919/DATE.2017.7927143.

Berardinelli, L., Eramo, R., Bruneliere, H., Gomez, A., Cicchetti, A., Said, B., Brosse, E., Herrera, F., Madi, G., Giner, J., Pandolfo, L., Pulina, L., Wimmer, M., Tisi, M., Saadatmand, M., Potena, P., König, S., Mutillo, V., Afzal, W., 2022. AIDOaRt D3.1 - Report on foundations of MDE and AIOPS for DevOps. https://www.aidoart.eu/aidoart/results/deliverables, (Accessed 14 April 2023).

Bergelin, J., Strandberg, P.E., 2022. Industrial requirements for supporting ai-enhanced model-driven engineering. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. pp. 375–379. http://dx.doi.org/10.1145/3550356.3561609.

Bersani, M.M., Marconi, F., Tamburri, D.A., Nodari, A., Jamshidi, P., 2019. Verifying big data topologies by-design : a semi-automated approach. J. Big Data 6, 40.

Blumreiter, M., Greenyer, J., Garcia, F.C., Klos, V., Schwammberger, M., Sommer, C., Vogelsang, A., Wortmann, A., 2019. Towards self-explainable cyber–physical systems. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion. MODELS-C, IEEE Computer Society, Los Alamitos, CA, USA, pp. 543–548.

Bonaccorso, G., 2017. Machine Learning Algorithms. Packt Publishing Ltd.

Bordeleau, F., Cabot, J., Dingel, J., Rabil, B.S., Renaud, P., 2020a. Towards modeling framework for devops: Requirements derived from industry use case. In: Bruel, J.-M., Mazzara, M., Meyer, B. (Eds.), Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. Springer International Publishing, Cham, pp. 139–151.

Bordeleau, F., Combemale, B., Eramo, R., van den Brand, M., Wimmer, M., 2020b. Towards model-driven digital twin engineering: Current opportunities and future challenges. In: ICSMM 2020. Springer, pp. 43–54.

Brambilla, M., Cabot, J., Wimmer, M., 2017. Model-Driven Software Engineering in Practice, second ed. Morgan & Claypool Publishers, U.S.A.

Bruneliere, H., de Kerchove, F.M., Daniel, G., Madani, S., Kolovos, D., Cabot, J., 2020. Scalable model views over heterogeneous modeling technologies and resources. Softw. Syst. Model. 19 (4), 827–851.

Bruneliere, H., Muttillo, V., Eramo, R., Berardinelli, L., Gomez, A., Bagnato, A., Sadovykh, A., Cicchetti, A., 2022. Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber–physical systems. Microprocess. Microsyst. 94, 104672.

Bruneliere, H., Perez, J.G., Wimmer, M., Cabot, J., 2015. EMF Views: A view mechanism for integrating heterogeneous models. In: 34th International Conference on Conceptual Modeling. ER 2015, Springer, pp. 317–325.

Burgueño, L., Kessentini, M., Wimmer, M., Zschaler, S., 2021. MDE intelligence 2021: 3rd workshop on artificial intelligence and model-driven engineering. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS 2021 Companion, Fukuoka, Japan, October 10-15, 2021. IEEE, pp. 148–149. http://dx.doi.org/10.1109/MODELS-C53483.2021.00026.

Castellanos, C., Pérez, B., Correal, D., Varela, C.A., 2020. A model-driven architectural design method for big data analytics applications. In: 2020 IEEE International Conference on Software Architecture Companion. ICSA-C, IEEE, U.S.A, pp. 89–94.

Castellanos, C., Varela, C.A., Correal, D., 2021. Accordant: A domain specific-model and devops approach for big data analytics architectures. J. Syst. Softw. 172, 110869. http://dx.doi.org/10.1016/j.jss.2020.110869, URL https://www.sciencedirect.com/science/article/pii/S0164121220302594.

Charley Rich, S.G., Pankaj Prasad, 2019a. Market Guide for Aiops Platforms, Id G00378587. Tech. Rep., Gartner Research.

Charley Rich, S.G., Pankaj Prasad, 2019b. Market guide for AIOps platforms, ID G00378587. https://www.ibm.com/downloads/cas/AXO20DXM, (Accessed 06 December 2018).

Colantoni, A., Berardinelli, L., Wimmer, M., 2020. Devopsml: Towards modeling devops processes and platforms. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. pp. 1–10.

Combemale, B., Wimmer, M., 2019. Towards a model-based devops for cyber–physical systems. In: Second International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. Vol. 12055, Springer, Germany, pp. 84–94.

Dang, Y., Lin, Q., Huang, P., 2019. Aiops: Real-world challenges and research innovations. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings. ICSE-Companion, pp. 4–5.

Dehghani, M., Berardinelli, L., Wimmer, M., 2020. Towards modeling process mining for graphical editors. In: 26th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. pp. 929–933.

Derler, P., Lee, E.A., Sangiovanni Vincentelli, A., 2012. Modeling cyber–physical systems. Proc. IEEE 100 (1), 13–28. http://dx.doi.org/10.1109/JPROC.2011.2160929.

Desfray, P., 2015. Model repositories at the enterprises and systems scale: The modelio constellation solution. In: 2015 International Conference on Information Systems Security and Privacy. ICISSP, IEEE, U.S.A, pp. IS–17.

Di Sipio, C., Di Rocco, J., Di Ruscio, D., Nguyen, P.T., 2023. MORGAN: a modeling recommender system based on graph kernel. Softw. Syst. Model..

Ebert, C., Gallardo, G., Hernantes, J., Serrano, N., 2016. DevOps. IEEE Softw. 33 (3), 94–100. http://dx.doi.org/10.1109/MS.2016.68.

Eramo, R., Bordeleau, F., Combemale, B., van den Brand, M., Wimmer, M., Wortmann, A., 2022. Conceptualizing digital twins. IEEE Softw. 39 (2), 39–46.

Eramo, R., Muttillo, V., Berardinelli, L., Bruneliere, H., Gomez, A., Bagnato, A., Sadovykh, A., Cicchetti, A., 2021. Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber–physical systems. In: 2021 24th Euromicro Conference on Digital System Design. DSD, IEEE, U.S.A, pp. 303–310.

Eramo, R., Said, B., Oriol, M., Bruneliere, H., 2023. Dataset for model-based and intelligent automation in DevOps: the AIDOaRt project's experience. In: The CSV Files Contain the Raw Data. the PDF File Contains the Tabular Data and the Generated Charts. http://dx.doi.org/10.5281/zenodo.7825177.

Felderer, M., Enoiu, E.P., Tahvili, S., 2023. Artificial intelligence techniques in system testing. In: Optimising the Software Development Process with Artificial Intelligence. Springer, pp. 221–240.

Fitzgerald, B., Stol, K.-J., 2017. Continuous software engineering: A roadmap and agenda. J. Syst. Softw. 123, 176–189.

Gartner, 2019. Gartner predicts the future of AI technologies.

Hofer, F., 2018. Architecture, technologies and challenges for cyber–physical systems in industry 4.0: A systematic mapping study. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM '18, pp. 1–10. http://dx.doi.org/10.1145/3239235.3239242.

Hugues, J., Hristosov, A., Hudak, J.J., Yankel, J., 2020. Twinops-devops meets model-based engineering and digital twins for the engineering of cps. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. pp. 1–5.

Jabbari, R., bin Ali, N., Petersen, K., Tanveer, B., 2016. What is devops? a systematic mapping study on definitions and practices. In: XP2016, XP '16 Workshops. ACM, New York, NY, USA, pp. 1–11.

Jahić, B., Guelfi, N., Ries, B., 2023. Semkis-dsl: A domain-specific language to support requirements engineering of datasets and neural network recognition. Information 14 (4), http://dx.doi.org/10.3390/info14040213, URL https://www.mdpi.com/2078-2489/14/4/213.

Kazman, R., Klein, M., Clements, P., 2000. Atam: Method for architecture evaluation.

Leite, L., Rocha, C., Kon, F., Milojicic, D., Meirelles, P., 2019. A survey of devops concepts and challenges. ACM Comput. Surv. 52 (6).

Maalej, W., Nayebi, M., Johann, T., Ruhe, G., 2016. Toward data-driven requirements engineering. IEEE Softw. 33 (1), 48–54.

Manjunath Bhat, C.R., 2019. Augment Decision Making in Devops using Ai Techniques, Id G00383246. Tech. Rep., Gartner Research.

Mayerhofer, T., Wimmer, M., Berardinelli, L., Drath, R., 2018. A model-driven engineering workbench for caex supporting language customization and evolution. IEEE Trans. Ind. Inform. 14, 2770–2779.

Muškardin, E., Aichernig, B.K., Pill, I., Pferscher, A., Tappler, M., 2021. Aalpy: An active automata learning library. In: Hou, Z., Ganesh, V. (Eds.), Automated Technology for Verification and Analysis. pp. 67–73.

Nigmatullin, I., Sadovykh, A., Ebersold, S., Messe, N., 2023. Rqcode: Security requirements formalization with testing. In: IFIP International Conference on Testing Software and Systems. Springer, pp. 126–142.

Russo, B., Jaatun, M.G., Abrahamsson, P., Botterweck, G., Ghanbari, H., Kettunen, P., Mikkonen, T.J., Mjeda, A., Münch, J., Duc, A.N., Wang, X., 2020. Towards a secure devops approach for cyber–physical systems: An industrial perspective. Int. J. Syst. Softw. Secur. Prot. (IJSSSP) 11 (2), 38–57. http://dx.doi.org/10.4018/IJSSSP.2020070103.

Sadovykh, A., Said, B., Truscan, D., Bruneliere, H., 2024. An iterative approach for model-based requirements engineering in large collaborative projects: A detailed experience report. Sci. Comput. Program. 232, 103047.

Sadovykh, A., Truscan, D., Bruneliere, H., 2021. Applying model-based requirements engineering in three large European collaborative projects: An experience report. In: 2021 IEEE 29th International Requirements Engineering Conference. RE, IEEE, U.S.A, pp. 367–377.

Said, B., Sadovykh, A., Brosse, E., Bagnato, A., 2022. Towards aidoart objectives via joint model-based architectural effort. In: RCIS Workshops. CEUR-WE.ORG, Online, pp. 1–6.

Schmidt, D.C., 2006. Guest editor's introduction: Model-driven engineering. Computer 39 (2), 25–31.

Sebastián, G., Gallud, J.A., Tesoriero, R., 2020. Code generation using model driven architecture: A systematic mapping study. J. Comput. Lang. 56, 100935.

Silva, S., Tuyishime, A., Santilli, T., Pelliccione, P., Iovino, L., 2023. Quality metrics in software architecture. In: 20th IEEE International Conference on Software Architecture, ICSA 2023, L'Aquila, Italy, March 13-17, 2023. IEEE, pp. 58–69. http://dx.doi.org/10.1109/ICSA56044.2023.00014.

Suryadevara, J., Tiwari, S., 2018. Adopting mbse in construction equipment industry: An experience report. In: APSEC 2018. IEEE, pp. 512–521.

Thompson, H., Reimann, M., Ramos-Hernandez, D., 2018. Platforms4CPS, Key Outcomes and Recommendations. Germany,

Törngren, M., Sellgren, U., 2018. Complexity challenges in development of cyber–physical systems. In: Principles of Modeling: Essays Dedicated To Edward a. Lee on the Occasion of His 60th Birthday. pp. 478–503.

Valente, G., Fanni, T., Sau, C., Di Mascio, T., Pomante, L., Palumbo, F., 2021. A composable monitoring system for heterogeneous embedded platforms. ACM Trans. Embed. Comput. Syst. 20 (5), http://dx.doi.org/10.1145/3461647.

van den Heuvel, W., Tamburri, D.A., 2020. Model-driven ml-ops for intelligent enterprise applications: Vision, approaches and challenges. In: Business Modeling and Software Design - 10th International Symposium, BMSD 2020, Berlin, Germany, July 6-8, 2020, Proceedings. In: Lecture Notes in Business Information Processing, Vol. 391, Springer, Germany, pp. 169–181.

Wan, Z., Xia, X., Lo, D., Murphy, G.C., 2021. How does machine learning change software development practices? IEEE Trans. Softw. Eng. 47 (9), 1857–1871. http://dx.doi.org/10.1109/TSE.2019.2937083.

Zampetti, F., Tamburri, D.A., Panichella, S., Panichella, A., Canfora, G., Penta, M.D., 2022. Continuous integration and delivery practices for cyber–physical systems: An interview-based study. ACM Trans. Softw. Eng. Methodol. http://dx.doi.org/10.1145/3571854.

Zhao, T., Huang, X., Cao, Y., 2017. Deepdsl: A compilation-based domain-specific language for deep learning. CoRR abs/1701.02284, arXiv:1701.02284. URL http://arxiv.org/abs/1701.02284.

Zhou, Z.-H., 2021. Machine Learning. Springer Nature.