Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

ARTICLE TYPE

AI-Enabled Software and System Architecture Frameworks: Focusing on Smart Cyber-Physical Systems (CPSs)

Armin Moin[1] | Atta Badii[2] | Stephan Günnemann[3] | Moharram Challenger[4]

[1]Department of Computer Science, University of Colorado Colorado Springs, Colorado, United States, Email: amoin@uccs.edu

[2]Department of Computer Science, University of Reading, United Kingdom, Email: atta.badii@reading.ac.uk

[3]School of Computation, Information, and Technology, and Munich Data Science Institute (MDSI), Technical University of Munich, Bavaria, Germany, Email: s.guennemann@tum.de

[4]Department of Computer Science, University of Antwerp and Flanders Make, Belgium, Email: moharram.challenger@uantwerpen.be

Correspondence
Armin Moin, Department of Computer Science, College of Engineering and Applied Science, University of Colorado Colorado Springs, 1420 Austin Bluffs Pkwy, Colorado Springs, CO 80918, United States
Email: amoin@uccs.edu

Present Address
Department of Computer Science, College of Engineering and Applied Science, University of Colorado Colorado Springs, 1420 Austin Bluffs Pkwy, Colorado Springs, CO 80918, United States

Abstract

Several architecture frameworks for software, systems, and enterprises have been proposed in the literature. They identified various stakeholders and defined architecture viewpoints and views to frame and address stakeholder concerns. However, the stakeholders with data science and Machine Learning (ML) related concerns, such as data scientists and data engineers, are yet to be included in existing architecture frameworks. Therefore, they failed to address the architecture viewpoints and views responsive to the concerns of the data science community. In this paper, we address this gap by establishing the architecture frameworks adapted to meet the requirements of modern applications and organizations where ML artifacts are both prevalent and crucial. In particular, we focus on ML-enabled Cyber-Physical Systems (CPSs) and propose two sets of merit criteria for their efficient development and performance assessment, namely the criteria for evaluating and benchmarking ML-enabled CPSs and the criteria for evaluation and benchmarking of the tools intended to support users through the modeling and development pipeline. This study deploys multiple empirical and qualitative research methods based on literature review and survey instruments, including expert interviews and an online questionnaire. We collect, analyze, and integrate the opinions of 77 experts from over 25 organizations in 10 countries to devise and validate the proposed framework.

KEYWORDS:
architecture frameworks, viewpoints, views, machine learning, cyber-physical systems, modeling, merit criteria, qualitative research

1 | INTRODUCTION

Architecture frameworks provide 'conventions, principles, and practices for the description of architectures established within a specific domain of application or community of stakeholders' [1]. There exist several well-established examples, including The Open Group Architecture Framework (TOGAF)[2,3], the Department of Defense Architecture Framework (DoDAF)[4], the Treasury Enterprise Architecture Framework (TEAF)[5], the British Ministry of Defence Architecture Framework (MODAF)[6], the Zachman Framework[7], the "4+1" View Model of Software Architecture[8], and the Reference Model of Open Distributed Processing (RM-ODP)[9,10,11,12]. TOGAF, DoDAF, TEAF, and MODAF were primarily concerned with enterprise architectures. MODAF was replaced by the NATO Architecture Framework (NAF), which in its fourth version (NAFv4)[13] provided guidance not only on describing enterprise architectures but also system architectures for military and business use. Furthermore, the Zachman

Framework, a generic framework for information system and enterprise architectures, and the "4+1" View Model of Software Architecture were among the early work that offered the foundations for the more recent architecture frameworks. Last but not least, RM-ODP was not merely a reference model but a set of four international standards, including an architecture framework for distributed information processing in heterogeneous environments. Each of the above-mentioned architecture frameworks identified stakeholders, such as end-users, software developers, system integrators, system engineers, business domain experts, executives, and other corporate functions[8,3]. Also, they defined architecture viewpoints and views. Section 2 provides some background on architectural artifacts, such as viewpoints and views.

However, prior work in architecture frameworks did not recognize any stakeholder with ML-related concerns, although ML is increasingly making software and information systems smart, and ML components are assuming a prominent role in many systems and organizations. Consequently, existing architecture frameworks lack any viewpoint or view dedicated to the stakeholders of ML artifacts. Hence, we argue that architecture descriptions of smart (i.e., ML-enabled) software systems, which need to reflect on their ML components and the interactions of the ML components with the non-ML components, cannot be adequately designed using state-of-the-art architecture frameworks. Essentially, ML is a separate field, a sub-discipline of Artificial Intelligence (AI) rather than Software Engineering (SE). Thus, it has its vocabulary, skill set, and know-how, which are different than the ones possessed by typical software developers. Recent research work, such as the interviews conducted by Nahar et al.[14], stressed the necessity of collaboration between software engineers and other specialists, such as data scientists, for building ML-enabled systems and the associated challenges. In particular, they emphasized the human factors of collaboration, including the need to separate the data science and SE work, as well as to coordinate between them, negotiate and document interfaces and responsibilities, and plan the system operation and evolution. According to them, those human collaboration challenges appeared to be the primary obstacles in developing ML-enabled systems. Additionally, past work mainly focused on ML models, such as the challenges of learning, testing, or serving ML models. They were rarely concerned with the entire system, with many non-ML parts into which the ML model is embedded as a component, which requires coordinating and integrating work from multiple experts or teams[14]. Similarly, Lewis et al.[15] elaborated on the so-called *ML Mismatch* problem that typically occurs in the process of development, integration, deployment, and operation of ML-enabled systems, due to incorrect assumptions made about system elements by different stakeholders, such as data scientists, software engineers, and the operations team.

In this paper, we postulate that ML artifacts of smart systems deserve to be treated as having a separate identity, distinguished from software source code and raw data. Moreover, stakeholders should see ML aspects of systems in different ways (i.e., using tailored notations and at various detail levels) such that they can better understand architecture descriptions of ML-enabled systems. In this way, they should become capable of efficient communication with other stakeholders and collaborate to contribute to the system or its architecture description, for example, by rigorous requirement elicitation and tracing. For instance, a data scientist is often in charge of analytics modeling and is primarily concerned with ML performance metrics, such as Accuracy, Precision, and Recall of the ML model when faced with unseen test data. However, a data engineer is responsible for analytics operations, thus ensuring scalable data processing. By contrast, other stakeholders, such as software engineers, software architects, database engineers, and system engineers, consider other aspects of performance that might be affected by the performance of the ML component, for example, by the delay introduced as a result of the predictions of the ML model, or might be unrelated to the ML component. The same holds for security: software and system engineers might underestimate the potential vulnerability of ML-enabled systems through adversarial attacks on the ML models. Therefore, data scientists and software engineers often have different notions of security.

Further, we are particularly interested in architecture frameworks in the context of Cyber-Physical Systems (CPSs). CPSs are complex systems that connect the physical world with the virtual world of cyberspace[16]. A modern car, an aircraft, a smart grid, a robot in a production line, or the computer system controlling a chemical process at a plant are examples of CPSs. If a CPS possesses any AI or cognitive capability, for example, through an ML component, it is called a smart (or ML-enabled) CPS. Further, a CPS might be connected to the Internet (Internet of Things, IoT), or might be isolated (e.g., due to security or privacy concerns). CPSs are typically cross-application domains, cross-technologies, and cross-organizations[17]. Therefore, they may considerably benefit from the *separation of concerns* that architecture frameworks can offer.

Additionally, it is vital to have objective measures for evaluating, comparing, and benchmarking CPSs, in particular ML-enabled ones. This is not trivial since different stakeholders care about different aspects of the system depending on their viewpoints. Since it is often not feasible to address all stakeholder concerns in a perfect manner, a system designer has to make compromises. However, system designers and other stakeholders should ideally be aware of the set of all possible merit criteria of CPSs and the trade-offs already at the time of requirements specification. The same holds for the modeling tools that can be

used for designing and creating those systems: system engineers should be able to select and deploy an appropriate modeling tool based on the merit criteria of CPS modeling tools.

The contribution of this paper is twofold: i) it identifies several stakeholders of modern software, systems, and enterprises who might have ML-related concerns. Moreover, it proposes new architecture viewpoints and views concerning the ML aspects. ii) Focusing on ML-enabled CPSs, it proposes two sets of merit criteria that can help benchmark and evaluate the following: a) ML-enabled CPSs; b) modeling tools for designing and creating ML-enabled CPSs.

The remainder of this paper is structured as follows: Section 2 provides a brief background, whereas Section 3 reviews the related work in the literature. Further, Section 4 elaborates on the research design and methodology. Moreover, Section 5 proposes new stakeholders, viewpoints, and views. In addition, Section 6 offers the merit criteria. Also, Section 7 points out possible threats to validity. Finally, Section 8 concludes and suggests future work.

2 | BACKGROUND

The *ISO/IEC/IEEE 42010:2011* standard for architecture descriptions in systems and software engineering [1] defined the *architecture* of a system as fundamental concepts or properties of the 'system in its environment embodied in its elements, relationships, and in the principles of its design and evolution' [1]. An *Architecture Description (AD)* is a work product that expresses an architecture. Any system *stakeholder* has various *concerns* regarding the System under Consideration (SuC), also known as the System of Interest (SoI), in relation to its environment. Concerns arise throughout the life cycle of the system from the system requirements, design choices, implementation, and operations. Performance, reliability, security, privacy, distribution, openness, evolvability, modularity, cost, and regulatory compliance are a number of examples of concerns [1,3]. Furthermore, *Separation of Concerns (SoC)*, which is a vital *design principle* in SE, can be applied on different layers of abstraction. At a lower level, it is interpreted as the modularity of the software system implementation, with information hiding and encapsulation in modules that have well-defined interfaces. However, at a higher level, it means describing the system architecture from the perspective of different sets of concerns (i.e., different *architecture viewpoints*). This is what an *architecture view* does. It is a work product that expresses 'the architecture of a system from the perspective of specific system concerns' [1]. Additionally, an *architecture viewpoint* is a work product that establishes 'the conventions for the construction, interpretation, and use of architecture views to frame specific system concerns]' [1]. In other words, a view is what a stakeholder can see, whereas a viewpoint is where the stakeholder is looking from (i.e., the perspective or template that determines what they should see). A viewpoint is generic and can be stored in a library for reuse. However, a view is specific to the architecture for which it is created [3]. For instance, Table 1 and Figure 1 illustrate a sample architecture viewpoint and its corresponding architecture view, respectively, based on the TOGAF architecture framework. The view describes the architecture of a CPS concerning the physical location of the sensors in the distributed system.

TABLE 1 A sample architecture viewpoint in TOGAF [3]

Element	Description
Stakeholders	Chief Technology Officer, system engineer, system integrator
Concerns	Show the top-level relationships between geographical sites and deployed sensors.
Modeling technique	Nested boxes diagram. Outer boxes = locations; inner boxes = sensors. The semantics of nesting = sensors deployed at the locations.

Further, the aforementioned standard [1] defined a *model kind* as a set of conventions for a type of modeling. An architecture viewpoint comprises one or more model kinds. Similarly, an architecture view comprises one or more architecture models. Additionally, each and every architecture viewpoint governs one architecture view; also, each and every model kind governs one architecture model. For instance, 'data flow diagrams, class diagrams, Petri nets, balance sheets, organization charts, and

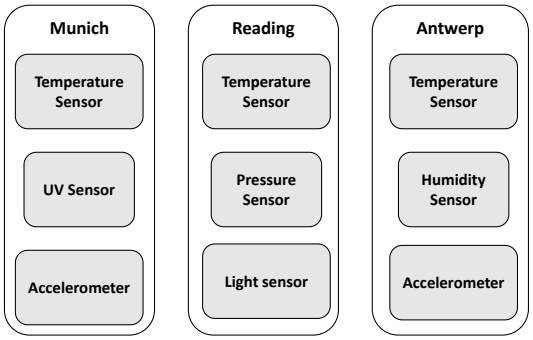

FIGURE 1 A sample architecture view in TOGAF[3] governed by the viewpoint of Table 1

state transition models' are model kinds[1]. Finally, *architecting* is the 'process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining, and improving an architecture throughout a system life cycle'[1]. It is typically conducted in the context of a project or an organization.

3 | STATE OF THE ART

In Section 1, we named a number of well-established architecture frameworks. The "4+1" View Model of Software Architecture[8], which was proposed in 1995, considers the following architecture viewpoints[1] for software systems: i) The **logical** viewpoint concentrates on the functionality of the software system. The key stakeholder group whose concerns are framed by this viewpoint is the end-user group. Moreover, the components in the views associated with this viewpoint are often classes. ii) The **process** viewpoint focuses on the non-functional aspects, such as performance, availability, fault tolerance, integrity, and scalability. This viewpoint is considered primarily for system integrators and system engineers. Furthermore, the components in the views associated with this viewpoint are typically tasks. iii) The **development** viewpoint is concerned with the software development and management aspects, such as the subsystems and modules' organization, reuse, and portability. The main stakeholder groups here are the programmer group (i.e., software developers), as well as software managers. Also, the components in the views associated with this viewpoint are usually modules and subsystems. iv) The **physical** viewpoint deals with scalability, performance, and availability on the physical layer, thus considering the network communications, distribution, topology, and the mapping of software onto the hardware. The main stakeholder group here is the system engineer group. Moreover, the components in the views associated with this viewpoint are often nodes. v) The **scenarios** viewpoint ensures understandability and includes a small set of use case scenarios that can show how the elements of the above-mentioned viewpoints can work together. In fact, this viewpoint is an abstraction of the system requirements. It is worth mentioning that this viewpoint is in some sense redundant from the other ones. This is why the name contains '+1'. However, it serves the important purpose of being a driver to discover the architectural elements during the design phase and playing a validation and an illustration role after the completion of the design for the test of an initial system prototype. This viewpoint is mainly intended for end-users and software developers. Lastly, the components of the views associated with this viewpoint are steps and scripts.

The "4+1" View Model of Software Architecture as explained above was an early but leading work concerned with the architecture of software systems. Other architecture frameworks, such as the Zachman Framework[7], which preceded this, or the other ones which succeeded it (see Section 1), had also identified a number of stakeholders, as well as several viewpoints and views. In the following, we refer to DoDAF[4] and TOGAF[2,3] whose latest versions were published in 2010 and 2018, respectively, as two representative examples. In particular, we highlight their data-related aspects since those are the most related ones to the target of this study.

DoDAF[4] defined several viewpoints including the Data and Information Viewpoint 1 (DIV-1, called **conceptual data** model), DIV-2 (called **logical data** model), and DIV-3 (called **physical data** model). DIV-1 was mainly intended for non-technical stakeholders and showed the high-level data concepts and their relationships. Relationships at this level would be simple (i.e., not attributed). However, DIV-2 bridged the gap between the conceptual and physical levels by introducing the attributes and

[1]Note that they used the term view in their work, which corresponded to the notion of viewpoint as defined by ISO/IEC/IEEE 42010:2011[1].

structural business process (activity) rules that formed the data structure. This viewpoint was intended for system architects and analysts. For example, an Entity-Relationship (ER) diagram or a Class diagram could be deployed as a candidate model kind for this viewpoint. Lastly, DIV-3 provided the physical schema for the data and information and was close to the actual implementation level. This could often be represented by tables, records, and keys in a relational database management system or through objects in an object-oriented data model. The physical data model (i.e., DIV-3) was intended for database engineers, software developers, and system engineers.

Moreover, the TOGAF standard[2,3] is closely related to the Zachman Framework[72]. In its 2006 version (i.e., v8.1.1), TOGAF[2] adopted the data flow viewpoint[3], which was intended for database engineers, 'concerned with the storage, retrieval, processing, archiving, and security of data', thus 'assuring ubiquitous access to high-quality data'[2]. Further, the latest version of TOGAF (v9.2) from 2018[3] promoted a set of data-related viewpoints (which they called data *diagrams*): i) The **conceptual data** viewpoint had the key purpose of depicting the relationships between the critical data entities of an enterprise. This viewpoint was intended for business stakeholders. ER diagrams or simplified UML Class diagrams could be deployed as the model kinds supporting this viewpoint. ii) The **logical data** viewpoint was intended for application developers and database designers and showed the logical views of the relationships between the critical data entities within an enterprise. iii) The **data dissemination** viewpoint illustrated the relationship between data entity, business service, and application components. It showed how the logical entities were physically realized by application components. iv) The **data security** viewpoint intended to depict which actor (i.e., person, organization, or system) has access to which enterprise data. This could be demonstrated in a matrix form or as a mapping. Moreover, it could show compliance with data privacy laws and other applicable regulations. v) The **data migration** viewpoint showed the flow of data between the source and target applications when implementing a package or packaged service-based solution, for example, if a legacy application was to be replaced. Packages might have their own data model. Thus, data transformation might be necessary. The transformation could include data quality processes, namely data cleansing, matching, merging, and consolidating data from different sources, as well as source-to-target mappings. This viewpoint could be deployed for data auditing and establishing traceability. The detail level of the supporting views could vary. vi) The **data life cycle** viewpoint enabled managing business data throughout their life cycle from conception until disposal. Each change in the state of data should be shown through the views supporting this viewpoint. The data were considered entities in their own rights, decoupled from business processes and activities. This allowed common data requirements to be identified.

4 | RESEARCH DESIGN AND METHODOLOGY

Due to the novelty of the idea of enhancing architecture frameworks to address ML, and the limited number of prior work on the merit criteria of CPSs and their modeling tools, we decided to carry out a qualitative study. While literature review surveys, systematic literature reviews, and meta-analyses are valuable, we lack sufficient qualitative, empirical studies which offer expert visions and open up new horizons for software engineering research. In this paper, we collected, analyzed, and synthesized the opinions of 77 subject matter experts from more than 25 organizations in over 10 countries[4] (Belgium, Canada, Denmark, France, Germany, Serbia, Switzerland, Turkey, the United Kingdom, and the United States) to devise and validate the proposed framework.

We conducted research in four steps: 1) Based on an initial literature review, we identified the gap in the literature and postulated the Research Questions (RQs) below. 2) Through the first round of expert interviews, we validated the identified gap and gained insights on designing our survey questionnaire for reaching out to a larger group of subject matter experts. 3) We conducted a survey study to answer the RQs and fill in the above-mentioned literature gap. 4) Finally, we concentrated on a certain group of complex systems that can considerably benefit from the abstraction provided by architecture frameworks, namely ML-enabled CPSs, and proposed merit criteria for evaluating and benchmarking these systems, as well as modeling tools for designing and creating such systems. We performed this using our own knowledge, literature review, and the second round of expert interviews. Below, we elaborate on these steps.

[2]In fact, there exists a mapping[2] between the TOGAF Architecture Development Method (ADM) and the Zachman Framework.
[3]Note that they used the term view in their work, which corresponded to the notion of viewpoint as defined by ISO/IEC/IEEE 42010:2011[1].
[4]The location of current employment of each expert was taken into account.

Step 1

We reviewed the related work in architecture frameworks as described in Sections 1 and 3. It transpired that none of the prior work had considered any ML aspects, such as stakeholders with ML-related concerns or architectural artifacts (e.g., viewpoints and views) regarding the ML aspects of smart systems, software, and enterprises. However, we contend that ML artifacts should be distinguished from raw data and software; moreover, considering their prevalence and crucial role in modern software, systems, and enterprises, their stakeholders must be identified in architecture frameworks and architecture descriptions should not ignore them. In other words, architecture frameworks must provide the required conventions, principles, and practices for describing the ML components of smart systems and the interrelation of the ML components with the non-ML components. Most importantly, architecture viewpoints and views that can frame and address the concerns of the identified stakeholders with ML-related concerns should be defined. Therefore, we studied the following RQs: **RQ1)** What are the recently emerged stakeholder groups of modern software, systems, and enterprises with ML-related concerns? **RQ2)** What are the viewpoints and views that should be devised in order to frame and address these stakeholders' concerns? **RQ3)** What are the merit criteria for ML-enabled CPSs? **RQ4)** What are the merit criteria for modeling tools supporting the design and creation of ML-enabled CPSs?

Step 2

This step constituted the first round of one-on-one expert interviews in the study. We interviewed 4 experts from the authors' networks from May to June 2021. The interviewee selection was based on convenience sampling (i.e., not random). The interviewee profiles were as described in Table 2 . The numbers in parentheses denote the frequencies.

TABLE 2 Summary of the demographic information of the interview partners in the first round of expert interviews

Type	Breakdown
Sex or gender (4)	female (1), male (3), other (0), no answer (0)
Age group (4)	below 18 (0), 18-24 (0), 25-39 (4), 40-60 (0), 60 plus (0), no answer (0)
Highest degree (4)	Bachelor's (1), Master's (1), Ph.D. (2), No academic degree (0), Other (0), no answer (0)
Field of expertise (4)	ML & SSE (1), Model-Driven Engineering, abbreviated as MDE (1), MDE & CPSs (1), general SSE (1)
Job or occupation (4)	researcher (1), senior software engineer (1), sales software engineer (1), data scientist (1)

Step 3

We carried out a survey study from July to September 2021 to answer RQ1 and RQ2 above, and fill in the aforementioned literature gap. The survey questionnaire was offered through a link (URL) as an online[5], self-administered survey, with the option of fully anonymous participation in the study. However, we collected the IP addresses to prevent any possible redundant participation. The questionnaire had four sections and a total of 25 questions. The order of the questions in each section and the order of the choices in the case of multiple-choice questions would be set on a random basis for each participant. For the survey design, we deployed some of the methods and practices recommended by Kitchenham and Pfleeger[19], as well as Bourque and Fielder[20]. For instance, we put the demographic questions in the last section. We had a total of 121 participants, out of which 60 participants did not answer the questionnaire at all. Therefore, we took the results of the remaining 61 participants into account. The selection process for the invitation of the subject matter experts to participate in this study was again based on convenience

[5]We used the open-source LimeSurvey software[18] on our own server.

sampling (i.e., not random) through the authors' networks, for example, by direct invitation of peers via email and sharing the URL on LinkedIn. Part of the participants' demographic information is summarized in Table 3 (the numbers in parentheses denote the frequencies). The average participation time was 14 minutes, whereas the median was 11 minutes. All questions were optional to answer except for the two questions regarding the consent of the survey participants with respect to their anonymity and receiving a pre-print of the study in the future.

TABLE 3 Summary of the survey participants' demographic information

Type	Breakdown
Sex or gender (61)	female (5), male (31), other (0), no answer (25)
Age group (61)	below 18 (0), 18-24 (2), 25-39 (30), 40-60 (6), 60 plus (1), no answer (22)
Highest degree (61)	Bachelor's (3), Master's (21), Ph.D. (17), No academic degree (0), Other (0), no answer (20)
DEA and ML expertise (61)	beginner (6), medium level (18), expert (16), no self estimation (i.e., don't know) (2), no answer (19)
SE expertise (61)	beginner (7), medium level (14), expert (18), no self estimation (i.e., don't know) (2), no answer (20)
Job or occupation (61)	data scientist & ML engineer (12), data engineer (1), software engineer (6), software architect (4), system engineer (4), data science & ML researcher (3), SE researcher (2), CS student (1), software community manager (1), CTO (1), software tester (1), no answer (25)

Step 4

Finally, we used our own knowledge and prior work in the literature (e.g., see [21,17,16]) to collect two sets of merit criteria for ML-enabled CPSs, as well as modeling tools that support the design and creation of such systems. We then conducted 12 one-on-one expert interviews from April to August 2022 to validate the devised merit criteria framework and extend or adapt it if applicable. In this way, we answered RQ3 and RQ4 above. Similar to the previous round, we deployed convenience sampling and selected the interview partners from the authors' networks. Table 4 illustrates the summary of their demographic information. Again, frequencies are denoted by the numbers in parentheses.

5 | ENHANCING ARCHITECTURE FRAMEWORKS

In this section, we enhance architecture frameworks by proposing new stakeholders, viewpoints, and views, as explained below.

5.1 | Identified stakeholders

In the following, we present the list of identified stakeholder groups for modern systems, software, and enterprises. In particular, we concentrate on the recently emerged ones who may have ML-related concerns, such as data scientists and data engineers.

TABLE 4 Summary of the demographic information of the interview partners in the second round of expert interviews

Type	Breakdown
Sex or gender (12)	female (1), male (11), other (0), no answer (0)
Age group (12)	below 18 (0), 18-24 (0), 25-39 (5), 40-60 (7), 60 plus (0), no answer (0)
Highest degree (12)	Bachelor's (0), Master's (2), Ph.D. (10), No academic degree (0), Other (0), no answer (0)
Field of expertise (12)	ML (3), CPSs (3), MDE (2), MDE & CPSs (2), general SSE (2)
Job or occupation (12)	professor (6), researcher (2), industrial practitioner (1), company co-founder (2), project manager (1)

According to the literature review reported in Sections 1 and 3, we name the following stakeholder groups that were already considered in prior architecture frameworks: 1) end-users, 2) business stakeholders, 3) database designers and engineers, 4) software architects and engineers (i.e., developers), 5) system designers, engineers, and integrators. Additionally, we believe that 6) network engineers and 7) security experts should be distinguished from system engineers and software engineers, respectively, given the sophisticated level of knowledge and skills that is required for designing and managing secured, pervasive technologies of modern systems and organizations. Furthermore, we noticed the stakeholder groups below during the first round of expert interviews: 8) safety and regulatory compliance engineers, 9) data protection (privacy) officers, and 10) ethics committees or boards. Also, the online survey participants pointed out the following stakeholder groups: 11) quality assurance (test) engineers, and 12) maintenance managers. Last but not least, we propose counting 13) data scientists (including ML engineers) and 14) data engineers among the stakeholders of modern systems, software, and enterprises, which often contain ML components or ML-enabled (sub-)systems. The proposed stakeholder groups were validated through the first round of expert interviews and the online survey.

Data scientists are responsible for analytics modeling. In fact, the task of developing methods for building efficient Data Analytics (DA) models (including ML models) to enable systems that can analyze data and *learn* from data lies at the core of data science. However, there is also a need for technologies regarding the deployment of DA models in products, services, and operational systems. This part is known as analytics operations [22]. Data engineers are typically concerned with this part, which is also called Data Engineering (DE). Together, DA (i.e., data science) and DE are called Data Engineering and Analytics (DEA) [23] or Data Science and Engineering (DSE) [24]. The typical workflow comprises analytics modeling (e.g., training ML models) by data scientists, followed by the integration and deployment of the data analytics artifacts (e.g., the trained ML models) in the data analytics and ML components, as well as in the larger systems (e.g., ML-enabled CPSs) by data engineers in collaboration with software engineers, database engineers, system engineers, etc. Afterward, the system should be handed over to the operations team [15].

Lastly, it is clear that DA models should not be confused with data models, datasets, or data instances. For instance, a DA (e.g., ML) model can be a Probabilistic Graphical Model (PGM), an Artificial Neural Network (ANN), or a Hidden Markov Model (HMM), which may enable inference on data. By contrast, raw data in datasets or data streams have a different nature. For example, one can use raw data to *train* an ML model. Also, a data model refers to an abstract model of the entities and their relationships in a database. Therefore, data models must be distinguished from DA models. Similarly, data scientists and data engineers should not be confused with database engineers.

5.2 | Proposed viewpoints and views

We propose two new architecture viewpoint categories to frame the concerns of data scientists and data engineers in architecture frameworks of systems, software, and enterprises. We call the new viewpoint categories analytics modeling (alternatively DA or

data science) and analytics operations (or DE), respectively. Moreover, we propose adopting and adapting existing notations and model kinds to realize corresponding views for the new viewpoints, as well as new views for the viewpoints of other stakeholders communicating and collaborating with data scientists and data engineers.

According to our own knowledge about the state of practice in the field of DEA (or DSE), we envisioned the mathematical notations commonly used in DA and ML, as well as the graphical notation of PGMs[25], and the Data-Flow Graphs (DFGs), also known as Computational Graphs (CGs)[26] for ANNs, as potentially adequate candidates for providing the necessary model kinds that should serve the architecture views required for supporting the proposed new architecture viewpoints. In addition, we assumed that existing UML diagrams could be adopted for adapting the viewpoints of other stakeholders in order to enable their appropriate communication and collaboration with data scientists and data engineers. The empirical study through the survey questionnaire confirmed these assumptions and helped us devise the new viewpoints and views using existing notations and model kinds, as illustrated in Table 5 . The list of notations and model kinds in each row of the table is ordered based on the opinions of the survey participants concerning the suitability of each option for the specific purpose. The radar diagrams in Figures 2 and 3 show this for the collaboration of data scientists with their peers, whereas the ones in Figures 4 and 5 demonstrate this for the collaboration of data engineers with their peers. In fact, we asked the same question from each stakeholder group in two different ways: what is the best model kind and notation for their collaboration; and what is the most suitable model kind and notation for describing the system architecture from their viewpoint?

TABLE 5 Notations and model kinds supporting the viewpoints and views of ML-enabled architecture frameworks for framing and addressing stakeholder concerns with respect to ML

Stakeholders communicating or collaborating	**Notations and model kinds supporting the viewpoints and views**
Data scientists with their peers	i) mathematical notations, ii) charts, diagrams, or plots, iii) DFGs or CGs[26], iv) PGMs[25], v) data flow diagrams (or UML activity diagrams showing the flow of data rather than the flow of control), for example, for the data analytics pipeline
Data engineers with their peers	i) data flow diagrams, ii) UML class diagrams, iii) DFGs or CGs[26], iv) Entity-Relationship (ER) diagrams, v) mathematical notations
End-users with data scientists and engineers	i) text documents, ii) charts, diagrams, or plots, iii) tables or matrices, iv) data flow diagrams, v) UML use case diagrams
Business stakeholders with data scientists and engineers	i) charts, diagrams, or plots, ii) text documents, iii) tables or matrices, iv) UML use case diagrams
Database designers and engineers with data scientists and engineers	i) ER diagrams, ii) UML class diagrams, iii) data flow diagrams, iv) UML use case diagrams, v) tables or matrices
Software architects and engineers with data scientists and engineers	i) UML class diagrams, ii) data flow diagrams, iii) UML use case diagrams, iv) ER diagrams
System designers, engineers, integrators, and network engineers with data scientists and engineers	i) UML deployment diagrams, ii) data flow diagrams, iii) DFGs or CGs[26] augmented with physical (i.e., deployment) information, iv) UML class diagrams
Security experts with data scientists and engineers	i) data flow diagrams, ii) UML deployment diagrams, iii) DFGs or CGs[26] augmented with physical (i.e., deployment) information, iv) ER diagrams, v) mathematical notations, vi) UML class diagrams
Safety and regulatory compliance engineers, data protection (privacy) officers, and ethics committees or boards with data scientists and engineers	i) text documents, ii) data flow diagrams, iii) ER diagrams, iv) DFGs or CGs[26] augmented with physical (i.e., deployment) information, v) tables or matrices, vi) UML deployment diagrams

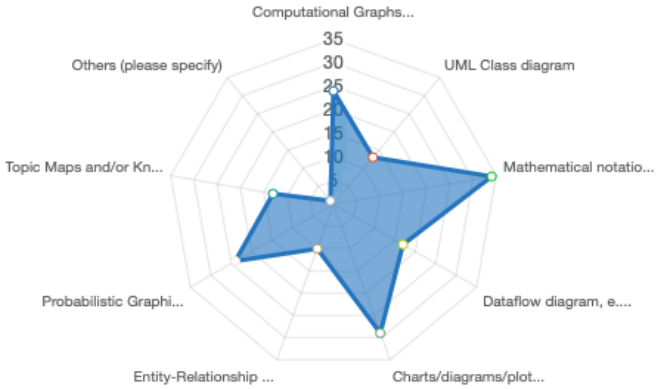

FIGURE 2 Model kinds and notations for the collaboration of data scientists with their peers. Choices (clockwise): a) Computational Graphs (CG), also known as Data-Flow Graphs (DFG). b) UML class diagrams. c) Mathematical notation showing the mathematical model (e.g., probability distributions, as well as objective or loss function). d) Data flow diagrams (e.g., using the UML Activity diagram notation) showing the upstream and downstream components in the pipeline. e) Charts, diagrams, or plots (e.g., histograms, pie charts, and scatter plots). f) Entity-Relationship (ER) diagrams. g) Probabilistic Graphical Models (PGMs). h) Topic maps, knowledge graphs (e.g., RDF graphs), or Ontologies. i) Others (please specify).

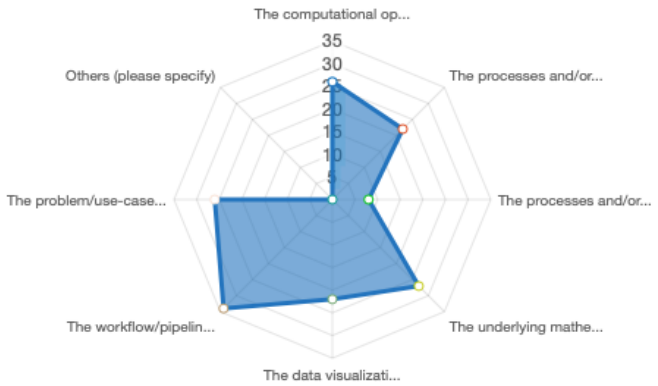

FIGURE 3 Architecture description from the perspective of data scientists. Choices (clockwise): i) The computational operations and the flow of data among them. ii) The processes or components, and the flow of data among them. iii) The processes or components, and the flow of control among them. iv) The underlying mathematical model. v) The data visualization. vi) The workflow or pipeline for data analytics and machine learning. vii) The problem (use case) domain concepts and their relationships. viii) Others (please specify).

6 | MERIT CRITERIA

In this section, we propose two sets of merit criteria for evaluating, comparing, and benchmarking ML-enabled CPSs and modeling tools supporting their design and creation. We grouped the closely related criteria in each part (e.g., security and privacy). However, there is no particular order for the presented merit criteria groups in the lists below.

6.1 | Merit criteria for smart CPSs

We categorized the smart CPS merit criteria into three groups to enable discussions with experts in the domains that were most relevant to their field of expertise: i) general SSE merit criteria (which also hold for CPSs); ii) merit criteria related to

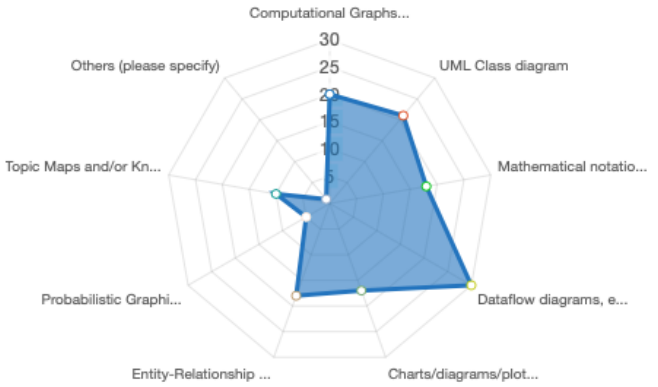

FIGURE 4 Model kinds and notations for the collaboration of data engineers with their peers. The choices were the same as in Figure 2 .

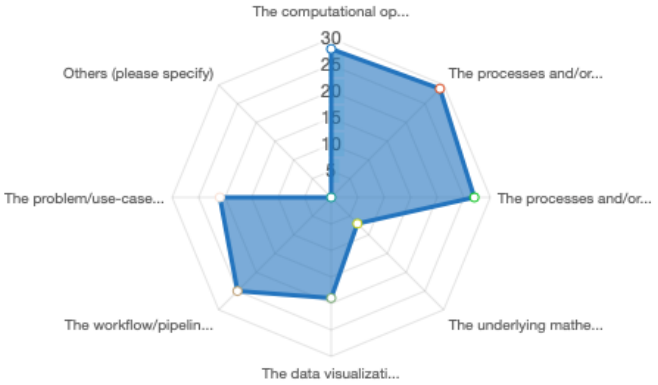

FIGURE 5 Architecture description from the perspective of data engineers. The choices were the same as in Figure 3 .

distributed computing, CPSs, and the IoT; iii) merit criteria pertaining to DA (including ML) and DE (i.e., DEA). However, it is acknowledged that the border between the first and the second categories might not always be clear. Despite that, we decided to keep the categorization as it was, in order to achieve a modular and reusable framework.

Under each category, a set of merit criteria had already been collected based on insights arising from the initial set of survey interviews as well as the results of previous research (e.g., see[21,17,16]). Before making the minds of the interviewees biased with our input, we asked them based on what criteria they would evaluate a system, a software, a CPS, an ML component of a system, or a modeling tool (whichever is applicable to their field of specialty) from their perspective. The lists of merit criteria in the remainder of this section reflect on both their and our inputs merged together. Nevertheless, the most vital merit criterion for every system is the fulfillment of its functional and non-functional requirements as specified by the stakeholders. Every stakeholder has a set of concerns. However, it is not feasible for the system designer to satisfy all of them. The system designer should convince the stakeholders that compromises have to be made. As mentioned in Section 1, our goal is to allow stakeholders to become aware of the full spectrum of the evaluation criteria, thus encouraging discussions and decisions on the required compromises and trade-offs.

General SSE merit criteria

1. **Security and privacy protection:** regulatory compliance by design (as applicable)[27].

2. **Providing high (adaptive) usability, accessibility, user acceptance, as well as social and environmental acceptability:** this includes sustainability considerations such as energy efficiency, carbon emissions minimizing, reusability, recyclability, and repurposability [28,29,27,30,31].

3. **Modularity, maintainability, and evolution support:** simplification by design and eliminating avoidable complexity [27,31,32].

4. **Dependability, reliability, trustworthiness, availability, and robustness of performance** [27,31,32].

5. **Efficiency, portability, scalability, and concurrency** [31].

6. **Expressivity, explainability, and transparency:** avoidance of black box solutions as much as possible, as required.

7. **Cost-effectiveness:** Cost-effective to procure and maintain (the latter may overlap with the maintainability criterion above) [27].

8. ***Learning* capability**: that is, ML-enabled.

Merit criteria related to distributed computing, CPSs, and the IoT

1. **Safety compliance:** since CPSs deal with the physical world and may interact with humans, they are often safety-critical [33,34].

2. **Semantic interoperability of heterogeneous systems and open system design:** as part of the so-called *cross-** CPS properties [17,35,16].

3. **Failure recovery handling and resilience:** fault-tolerance and graceful recovery from failure (e.g., an aircraft can fly further even if one engine fails and can somehow glide even if both engines fail). This is a child class of the so-called *self-** CPS properties; exhibited by reflexive design, as a parent merit criterion which subsumes others, such as self-adaptive, self-learning, self-optimizing, self-monitoring, self-auditing, self-diagnosing, self-healing, self-repairing, self-accountable, self-expressive, and explanation-giving capability [33,17,36,37].

4. **Scalability and latency:** to satisfy the network throughput requirements and real-time performance as applicable. IoT requires scalability at a much higher level.

Merit criteria pertaining to Data Engineering and Analytics (DEA)

From the data scientists' perspective, the merit of trained ML models or ML components of smart systems that deploy trained ML models is typically decided according to the following categories of metrics:

1. ***Target-based* metrics,** such as Accuracy, Precision, Recall (i.e., the true positive rate, also known as sensitivity in the case of binary classification), specificity (i.e., the true negative rate, also known as selectivity in the case of binary classification), and F1-measure for classification; error, for example, Mean-Squared Error (MSE) for regression.

2. **High-speed performance** in terms of time (e.g., for making predictions).

3. **Generalization:** robustness of performance even with certain increased levels of noise in the datasets or changes in the data.

4. **Handling uncertainty:** the ability to detect that the data are different from the original data, (adversarial) attack detection, achieving high AUC (Area Under the Curve) scores (e.g., for the Receiver Operating Characteristic, abbreviated as ROC, curve, or for the Precision-Recall, abbreviated as PR, curve), and the quality of calibration of the ML model.

5. **End-to-end ML:** the capability of the ML model to capture the entire ML pipeline (including the pre-processing and post-processing tasks) and act as a *one-stop-shop*, thus offering the so-called end-to-end ML.

6. **Automated ML (AutoML):** the extent to which the ML modeling pipeline stages are (semi)-automated involving stages, such as data pre-processing (data cleaning and de-noising, dimensionality reduction, sparsity mitigation, re-scaling, standardization, normalization, re-sampling, and re-balancing), feature extraction, ML model selection, training, and hyperparameter tuning.

7. **Auto annotation:** capabilities to process data with variable annotation formats, machine-readable data as well as non-semantic data; (class-) labeled, unlabeled, and partially labeled data for supervised, unsupervised, and semi-supervised ML, respectively.

8. **Ethical and legal compliance by design:** this parent class merit criterion is particularly exhibited as privacy-preserving by design, dignity-preserving, handling algorithmic bias, inclusiveness by design, security by design (e.g., adversarial attacks resistance), environmental acceptability (expressed as energy efficiency and having a low carbon footprint).

9. **Explainability:** this distinguishes explainable by design ML models, such as PGMs from ML models that are not explainable by design, such as ANNs.

Further, from the data engineers' perspective, the merit of ML components of smart systems is typically assessed with respect to the metrics below:

1. **Scalable batch data processing or offline learning:** (i.e., coping with large or very large datasets in the order of 10-100 thousand, or 100 thousand to 1 million instances).

2. **Efficient stream processing or online learning:** Being capable of efficient stream processing or online learning (i.e., dealing with unbounded datasets); and possibly addressing continual or lifelong learning needs.

3. **Supporting transfer learning**.

4. **Disk space and memory efficiency:** in particular, in the case of resource-constrained devices, such as *TinyML* platforms[38].

5. **MLOps-enabled**: MLOps is the application of DevOps from SE to the ML domain.

6.2 | Merit criteria for CPS modeling tools

Our study concluded the following merit criteria for CPS modeling tools, most of which would be expected to apply to other modeling tools as well:

1. **domain-specific or domain-agnostic:** if domain-specific, then whether the domain of focus is a problem (use case) domain (e.g., healthcare) or a solution domain (e.g., cloud computing)? Domain-specific tools can often support a higher automation level and generate a higher quality code on average[39]. Moreover, if the domain of focus is a problem domain, the user of the tool should be a domain expert in that vertical (application) domain. In contrast, if the domain of focus is a solution domain, the user of the tool should be a practitioner familiar with that area.

2. **Suitability for the application domain:** the expressiveness of the modeling language deployed by the tool, the level of abstract syntax completeness and appropriateness (e.g., the meta-model containing the necessary concepts, relationships, and semantics)[40].

3. **Syntax usability:** (i.e., practitioner-friendly syntax) the adequacy and suitability of the vocabulary in the case of textual concrete syntax or the diagrams in the case of graphical concrete syntax.

4. **Modeling and development support:** Model management (e.g., versioning support), traceability, debuggability, documentation support, and testing support.

5. **Supporting multiple architecture viewpoints** (for various stakeholders)[41].

6. **Supporting collaborative modeling**.

7. **Supporting fully automated code generation:** end-to-end complete solution generation vs. only skeleton generation[40].

8. **Supported target platforms:** hardware architectures, operating systems, programming languages, and APIs supported for code generation.
9. **Support for model checking and formal verification**.
10. **Support for simulation at the design time**.
11. **Portability and support for web-based access** through web browsers.
12. **Interoperability** with other tools, APIs, and standards.
13. **Supported ML libraries,** frameworks, methods, algorithms, and techniques for code generation of ML components.
14. **Support for ML techniques for non-i.i.d[6] data,** such as sequential data (e.g., time series or DNA data).
15. **Certifiably and compliant of code generation:** assured standard compliance of generated code (e.g., as required for safety-critical context).
16. **Extensibility and adaptability** of the modeling language including the model transformations. Also, support for legacy systems.
17. **Open-source availability:** for example, permissive license (such as Apache, MIT, or BSD).
18. **Technical support:** as may be available through an open-source community or a company.
19. **User support tutorials:** the extent to which tutorials and examples are available for the users (and developers) as well as their effectiveness towards development efficiency.
20. **Performance leap:** increase in the performance of software designers and developers[42].
21. **Technology Readiness Level (TRL):** technical maturity of the tool.
22. **Code generation quality:** this includes a range of merit criteria sub-classes, such as functional correctness, efficiency, modularity, and elegance of the generated code[42,40].
23. **Report generation:** for example, this includes rendering plots and visualization of data to support data analytics.
24. **Support for systems integration and networking:** for example, support for selection of network topology or setting up logical connections (e.g., Virtual Private Networks, VPNs).
25. **Runtime support:** this includes support for asset monitoring and management, for example, support for Over Air Programming ("OTA") for sensors, and MODELS@RUNTIME[43].
26. **DevOps support:** the capability of working in a DevOps, for example, Continuous Integration (CI) setup.

7 | THREATS TO VALIDITY

Notwithstanding the limitations in this study arising from the difficulty to ensure large-scale participation and the concomitant constraints with respect to inclusion-exclusion criteria and the sampling choices that had to be made, it is evident that in achieving the elicitation of the opinions and insights of a relatively significant number of expert practitioners, the study has formalized a useful candidate set of merit criteria for community reflection in relation to the design trade-offs for smart CPSs and modeling tools. Below, we point out a number of potential internal and external threats to the validity of this study.

[6]The i.i.d. acronym stands for independent and identically distributed.

7.1 | Internal validity

First, our literature review was not systematic. We did not define any inclusion or exclusion criteria in advance. We reviewed the work that we were aware of and searched for more prior work on the Google search engine using related keywords, such as architecture framework, viewpoint, and machine learning. Second, our interviewees were interviewed through online video meetings in various settings (including different interviewers). Moreover, not all interview discussion topics and questions were the same. We deliberately matched the topics and questions to the experts' backgrounds and fields of expertise. Also, the expert interviews had various durations (30-60 minutes). The different settings might have affected the achieved results through the interviews. Last but not least, the survey questionnaire was not validated. Ideally, one should have selected a small number of participants in an initial validation phase and then conducted the main study using the validated questionnaire. To mitigate the risks, we consulted the experts in the first round of interviews to find the right questions.

7.2 | External validity

The sampled participant distribution may not be representative enough to make generalizations beyond this group. The total sampled population size is 77 which is rather a small number for making any generalization over the ML or SE communities. However, due to the limited resources and the scarcity of experts willing to dedicate their time to the study, we had to lower our expectations. Furthermore, for the same reason, we neither deployed a randomized method to select the study participants nor could we manage to have all stakeholder groups, roles, jobs, disciplines, or underrepresented populations in this field in academia and in the industry well represented here. For instance, some of the chosen experts had served as the advisors, advisees, or colleagues of some other participants in the study. Also, despite our endeavors to attract more female participants, we were not satisfied with the portion of self-reported female participants. However, one may argue that some of the participants who preferred not to answer the demographics question on sex or gender might have belonged to the minority group of female participants.

8 | CONCLUSION AND FUTURE WORK

In this paper, we have enhanced architecture frameworks to address ML-enabled software systems. We have identified the stakeholders who might have concerns with respect to the ML aspects, namely data scientists and data engineers. Moreover, we have proposed new architecture viewpoint categories (i.e., analytics modeling and analytics operations) as well as architecture views to frame and address the stakeholder concerns. In addition to the literature review, we have interviewed 4 subject matter experts and conducted an online survey with 121 participants (out of which 61 answered) to devise and validate the proposed framework.

Further, we have focused on a specific group of complex systems, which can particularly benefit from the separation of concerns and the abstraction provided by architecture frameworks, namely Cyber-Physical Systems (CPS), especially smart (i.e., ML-enabled) CPSs. We have collected two sets of merit criteria that could determine the qualification and the quality of such systems, as well as the merit of modeling tools used for creating them. In total, we have proposed 52 merit criteria groups for the mentioned systems and tools. To this aim, we have conducted the literature review, used our own knowledge, and interviewed 12 other experts.

The results of this study are expected to improve the effectiveness and increase the efficiency of software development projects, especially for ML-enabled CPSs, which require the communication and collaboration of various stakeholder groups with different backgrounds, concerns, metrics, and vocabularies from different domains, organizations, and regions. Objective measurement and validation of the intended performance leap would, however, require a thorough empirical, experimental study with practitioners working on specific case studies.

One limitation of the present study was the absence of particular groups of stakeholders, such as the operations team (see [15]), the maintenance managers, and quality assurance engineers. The two latter items were suggested by our online survey participants. In the future, these and other stakeholder groups can be studied. Also, the enhancement of architecture frameworks for other sub-disciplines of Artificial Intelligence (AI) beyond ML should be helpful. Furthermore, one can try to prioritize the proposed merit criteria, for example, depending on the context and situation. In the present study, we have not set any order for them. Additionally, we have presented a specific grouping of the merit criteria based on our own ideas and the experts' feedback.

However, future work may consider other variations for this classification. Last but not least, more concrete and measurable metrics and indicators for each of the merit criteria should be determined as part of future work.

DATA AVAILABILITY

To promote the open science policy and enable transparency to ensure the availability of the deployed research methods and the (anonymized) collected data for inspection and interpretation, we offer the supplementary material of this paper as open data on Zenodo: https://doi.org/10.5281/zenodo.7035572.

References

1. ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description. standard, ISO / IEC / IEEE; 2011.
2. The Open Group Architecture Framework (TOGAF) version 8.1.1. standard, The Open Group; 2006.
3. The Open Group Architecture Framework (TOGAF) version 9.2. standard, The Open Group; 2018.
4. The US Department of Defense Architecture Framework (DoDAF). standard, Department of the Defense; 2010.
5. The US Treasury Enterprise Architecture Framework (TEAF). standard, Department of the Treasury; 2000.
6. The British Ministry of Defence Architecture Framework (MODAF). standard, Ministry of Defence; 2012.
7. Zachman JA. A Framework for Information Systems Architecture. *IBM Systems Journal* 1987; 26(3).
8. Kruchten P. Architectural Blueprints — The "4+1" View Model of Software Architecture. *IEEE Software* 1995; 12(6): 42-50.
9. ISO/IEC 10746-1 - Information technology — Open Distributed Processing — Reference model: Overview. standard, ISO/IEC; 1998.
10. ISO/IEC 10746-4:1998 Information technology — Open Distributed Processing — Reference Model: Architectural semantics — Part 4. standard, ISO/IEC; 1998.
11. ISO/IEC 10746-2:2009 Information technology — Open distributed processing — Reference model: Foundations — Part 2. standard, ISO/IEC; 2009.
12. ISO/IEC 10746-3:2009 Information technology — Open distributed processing — Reference model: Architecture — Part 3. standard, ISO/IEC; 2009.
13. NATO ARCHITECTURE FRAMEWORK Version 4. standard, The North Atlantic Treaty Organization (NATO); 2018. https://www.nato.int/cps/en/natohq/topics_157575.htm?
14. Nahar N, Zhou S, Lewis G, Kästner C. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. In: ; 2022: 413-425
15. Lewis G, Bellomo S, Ozkaya I. Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems. In: ; 2021: 133-140
16. Geisberger E, Broy M., eds.*Living in a networked world. Integrated research agenda Cyber-Physical Systems (agendaCPS)*. acatech STUDYMunich, Germany: Herbert Utz Verlag . 2014.
17. Schaetz B. The Role of Models in Engineering of Cyber-Physical Systems – Challenges and Possibilities. In: CPS Week. ; 2014.
18. LimeSurvey. https://www.limesurvey.orghttps://www.limesurvey.org; . Accessed: 2022-04-13.

19. Kitchenham BA, Pfleeger SL. *Personal Opinion Surveys*: 63–92; London: Springer London . 2008

20. Bourque L, Fielder E. *How to conduct self-administered and mail surveys*. The survey kit ; 3Thousand Oaks: Sage . 1995.

21. Weyrich M, Klein M, Schmidt JP, et al. *Evaluation Model for Assessment of Cyber-Physical Production Systems*: 169–199; Cham: Springer International Publishing . 2017

22. Pivarski J, Bennett C, Grossman RL. Deploying Analytics with the Portable Format for Analytics (PFA). In: KDD '16. Association for Computing Machinery; 2016; New York, NY, USA: 579–588

23. TUM Data Engineering and Analytics Master's Program. https://www.tum.de/en/studies/degree-programs/detail/data-engineering-and-analytics-master-of-science-msc; n/a. Accessed on 2022-03-21.

24. Raj RK, Parrish A, Impagliazzo J, et al. An Empirical Approach to Understanding Data Science and Engineering Education. In: ITiCSE-WGR '19. Association for Computing Machinery; 2019; New York, NY, USA: 73–87

25. Bishop CM. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag . 2006.

26. Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Software available from tensorflow.org.

27. Lochmann K, Goeb A. A Unifying Model for Software Quality. In: WoSQ '11. Association for Computing Machinery; 2011; New York, NY, USA: 3–10

28. Badii A, Fuschi DL. User-Intimate Requirements Hierarchy Resolution Framework (UI-REF) in work-flow design for 3D media production & distribution. In: ; 2008.

29. Badii A, Fuschi D, Khan A, Adetoye A. Accessibility-by-Design: A Framework for Delivery-Context-Aware Personalised Media Content Re-purposing. In: Holzinger A, Miesenberger K., eds. *HCI and Usability for e-Inclusion*Springer Berlin Heidelberg; 2009; Berlin, Heidelberg: 209–226.

30. Naumann S, Kern E, Dick M, Johann T. Sustainable Software Engineering: Process and Quality Models, Life Cycle, and Social Aspects. In: Hilty LM, Aebischer B., eds. *ICT Innovations for Sustainability*Springer International Publishing; 2015; Cham: 191–205.

31. Wolski M, Walter B, Kupiński S, Chojnacki J. Software quality model for a research-driven organization—An experience report. *Journal of Software: Evolution and Process* 2018; 30(5): e1911. doi: https://doi.org/10.1002/smr.1911

32. Nistala P, Nori KV, Reddy R. Software Quality Models: A Systematic Mapping Study. In: ; 2019: 125-134

33. Badii A, Khan A, Raval R, et al. SITUATION ASSESSMENT THROUGH MULTI-MODAL SENSING OF DYNAMIC ENVIRONMENTS TO SUPPORT COGNITIVE ROBOT CONTROL. *Facta Universitatis, Series: Mechanical Engineering* 2014; 12(3): 251–260.

34. Sabaliauskaite G, Mathur AP. Aligning Cyber-Physical System Safety and Security. In: Cardin MA, Krob D, Lui PC, Tan YH, Wood K., eds. *Complex Systems Design & Management Asia*Springer International Publishing; 2015; Cham: 41–53.

35. Kunold I, Wöhrle H, Kuller M, Karaoglan N, Kohlmorgen F, Bauer J. Semantic Interoperability in Cyber-Physical Systems. In: . 2. ; 2019: 797-801

36. Vogel-Heuser B, Prieler J. Evaluation of selected metrics for flexibility of Cyber Physical Production Systems. In: ; 2017: 701-708

37. Karnouskos S, Ribeiro L, Leitão P, Lüder A, Vogel-Heuser B. Key Directions for Industrial Agent Based Cyber-Physical Production Systems. In: ; 2019: 17-22

38. Warden P, Situnayake D. *TinyML*. USA: O'Reilly Media, Inc. . 2019.

39. Kelly S, Tolvanen JP. *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley. 1st ed. 2008.

40. Challenger M, Kardas G, Tekinerdogan B. A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems. *Software Qual J* 2016; 24: 755-795. doi: https://doi.org/10.1007/s11219-015-9291-5

41. Giraldo FD, España S, Pastor Giraldo WJ. Considerations about quality in model-driven engineering. *Software Qual J* 2018; 26: 685–750. doi: https://doi.org/10.1007/s11219-016-9350-6

42. Santos F, Nunes I, Bazzan AL. Quantitatively assessing the benefits of model-driven development in agent-based modeling and simulation. *Simulation Modelling Practice and Theory* 2020; 104: 102126. doi: https://doi.org/10.1016/j.simpat.2020.102126

43. Fouquet F, Morin B, Fleurey F, Barais O, Plouzeau N, Jezequel JM. A Dynamic Component Model for Cyber Physical Systems. In: CBSE '12. Association for Computing Machinery; 2012; New York, NY, USA: 135–144

How to cite this article: A. Moin, A. Badii, S. Günnemann, and M. Challenger (2023), AI-Enabled Software and System Architecture Frameworks: Focusing on Smart Cyber-Physical Systems (CPSs), *Journal of Software: Evolution and Process (JSEP), Special Issue on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS)*, .