Applied Computing and Informatics

Design and evaluation of software reference architecture: a systematic mapping study

Sailesh Saras Chand
Fiji National University, Nasinu, Fiji
Bimal Aklesh Kumar
The University of Fiji – Saweni Campus, Lautoka, Fiji, and
Elisha Yumi Nakagawa
University of São Paulo, São Paulo, Brazil

Received 2 June 2026
Revised 25 July 2026
Accepted 25 July 2026

Abstract

Purpose – Design and evaluation of Software Reference Architecture (SRA) is a complex task. This paper aims to collate the overall design and evaluation strategies that have been in practice for the last 15 years. It oversees the publication trend, components of SRA design, knowledge sources, developed artifacts, and evaluation matrices. Moreover, it highlights the models or frameworks that guide the development and evaluation processes.

Design/methodology/approach – The study followed the systematic mapping methodology outlined by [8] after searching relevant titles from the Scopus database. Thereafter, data extraction was guided by three research questions: (1) What is the current state of research? (2) How are SRA's designed? and (3) How are SRA's evaluated? The authors selected 75 primary studies after applying the inclusion and exclusion criteria.

Findings – The mapping study reveals a steady growth in publications, with various methodologies used to design SRA and produce artifacts. Evaluation practices remain underexplored, with case studies and expert reviews as the primary methods. Frequently targeted quality attributes included modifiability, reusability, scalability, and performance. SRA research demonstrates increasing maturity; however, the field still lacks methodological consistency.

Originality/value – The paper collaborates design and evaluation perspectives from the academic and business community. It points to the salient facts that structure today's SRA design and evaluation status. This study provides a comprehensive overview of the field and outlines a research agenda to enhance the SRA design and evaluation process for future developers.

Keywords Reference architecture, Software reference architecture, Software architecture, Systematic review

Paper type Literature review

Introduction

Software Reference Architectures (SRA's) are reusable architectural frameworks that guide the development of software systems within an application domain [1–4]. Researchers argue that SRA's serve as architectural blueprints from which concrete system architectures can be instantiated, enabling organizations to ensure consistency and quality even when development is distributed across teams, locations, or organizational boundaries. Their use has expanded across sectors such as healthcare, automotive, finance, and critical systems. The well-established examples, such as CORBA in distributed computing, AUTOSAR in automotive software, and FHIR in healthcare information systems, demonstrate how SRAs can reduce design effort and streamline system integration in these domains [5, 6]. The benefits

Applied Computing and Informatics
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1108/ACI-05-2026-0344

commonly attributed to SRAs include reduced development cost, increased architectural consistency, improved interoperability, and enhanced maintainability [7].

Despite their relevance, research on how SRAs are designed and evaluated remains underexplored. Much of the existing literature focuses on emphasizing and classification. Thus, there is a need to consolidate current knowledge on the SRA design and evaluation process. To address the research gap, this study employs a systematic mapping study, a method that provides an overview of a research field by classifying and structuring existing literature. A systematic mapping approach is particularly appropriate in the context of SRAs, where research is conceptually diverse, spans multiple domains, and exhibits variation in design and evaluation practices. The guidelines proposed by Ref. [8] were used, as they are popular for conducting systematic mapping studies in the software engineering domain. A mapping study will help clarify the current state of the field and identify areas where empirical knowledge remains limited [9]. The objectives of the study are to characterize the evolution of SRA research over time, identify the knowledge sources that inform SRA design, examine the methods, frameworks, tools, and architectural artifacts used to design SRAs, and analyze the evaluation techniques, quality attributes, and applied metrics used to assess their effectiveness.

The key contributions of the systematic mapping study include: 1) providing an evidence-based synthesis of how SRAs are designed and evaluated, 2) offering a comprehensive classification of the design and evaluation approaches, and 3) setting a research agenda that aims at strengthening the SRA design and evaluation process. The remainder of this paper is structured as follows. The background section reviews the theoretical background and related research on SRA's. The methodology section outlines the design and execution of the mapping study. The reporting section presents the results of the classification and analysis. The discussion section presents the study's key findings and implications. Finally, the paper concludes with a summary of key insights and recommendations for future work.

Background

Software reference architecture

Software architecture defines the high-level structure, components, and interactions of software systems, outlining the design principles that guide their development [10, 11]. Over time, the field has diversified into several subdisciplines, including enterprise architecture, service-oriented architecture, and software reference architecture (SRA) [12, 13]. Among these, SRAs play a distinctive role by combining domain-specific knowledge with architectural best practices to guide the development of software systems within a particular domain [14]. SRA encapsulates domain knowledge and reusable design patterns in architectural layers, components, interface definitions, and communication protocols [15]. It provides a structured framework for building domain-specific software applications while ensuring architectural consistency and interoperability across multiple systems. For example, CORBA, AUTOSAR, SAP, and IBM's Cloud Reference Architecture serve as blueprints for deriving concrete architecture tailored to specific contexts. SRAs are designed to be instantiated and adapted to meet the specific requirements of individual projects, enabling developers to build systems that align with a shared architectural vision [16, 17].

SRA's offer a range of benefits that make them valuable assets in software engineering practice. According to Ref. [18], one of the primary advantages is reusability, as SRAs enable the reuse of common functionalities, architectural components, and design patterns across multiple software products within a domain. This reuse promotes consistency, reduces redundancy, and streamlines the development process. In addition, SRAs contribute to risk reduction, since they are typically derived from thorough analyses of domain requirements and proven design solutions, lowering the likelihood of system failure during implementation [19]. Another significant advantage is cost and time efficiency, as SRAs facilitate component reuse and standardization, thereby accelerating development cycles, minimizing rework, and shortening the time-to-market for new applications [20]. These benefits enhance software

quality, maintainability, and interoperability across diverse systems. However, adopting SRAs is not without challenges. One major limitation concerns the potential reduction of innovation, as software reference architectures often prescribe specific component choices and structural constraints that may restrict creative or unconventional design decisions [20], leading to architectural uniformity that hinders experimentation and domain-driven customization. Moreover, design flaws or suboptimal decisions made during the creation of an SRA can propagate to all derived systems, amplifying architectural weaknesses and leading to systemic inefficiencies [7, 21]. These challenges highlight the importance of rigorous design, continuous evaluation, and regular revision of SRAs to ensure they remain adaptable, relevant, and capable of supporting innovation while maintaining architectural coherence.

Related work

Secondary studies have been conducted in recent years to synthesize knowledge on SRAs across different domains. Most of these studies focus on specific application areas rather than providing a broad analysis of SRAs. For instance Ref. [22], surveyed architectural frameworks in edge computing, providing insights into industrial adoption trends. Similarly [23], Internet of Things (IoT) architectural styles, identifying recurring architectural solutions and design constraints. In another domain-specific effort [24], surveyed cyber reference architectures, focusing on their structural patterns and security implications. Beyond domain-focused reviews, a few studies have examined SRAs more generally. For example [25], context, goals, perspectives, and application domains. Table 1 summarizes the focus and contributions of existing secondary studies on reference architecture across different domains.

These studies collectively contribute valuable insights into how SRA's are defined. However, they tend to emphasize categorization and classification rather than exploring the processes underlying the design and evaluation of SRAs. The present study extends this line of research by adopting a more process-oriented perspective. Rather than focusing solely on categorization or domain-based characteristics, this study examines the end-to-end lifecycle of SRAs, including their design and evaluation. By integrating both design and evaluation dimensions, this systematic mapping provides a more comprehensive synthesis of SRA research, revealing methodological patterns, evidence gaps, and opportunities for future work.

Methodology

This study followed a systematic mapping study, which provides a structured and repeatable process for identifying, classifying, and visualizing the state of research within a software

Table 1. Summary of mapping studies conducted on SRA across various domains

Study	Domain	Focus of review	Key findings
[22]	Edge Computing	Architectural frameworks and adoption patterns	Highlights domain-specific architecture and architectural frameworks
[23]	Ambient Assisted Living	Reference architectures for assisted living systems	Identifies architectural needs for healthcare environments
[26]	Internet of Things	Mapping IoT architectural styles	Identifies recurring patterns and constraints
[24]	Cybersecurity	Cyber reference architectures	Highlights structural and security considerations
[5]	Cross-domain (Big Data ecosystems)	Systematic review of Big Data Reference Architectures	Identifies core architectural components and recurring challenges
[25]	General SRA landscape	Conceptual classification of SRAs	Categorizes SRAs by context, goals, and perspectives

engineering domain. Systematic mapping focuses on how research is distributed across publication outlets, topics, domains, methods, and evaluation strategies. The mapping process consisted of the following stages: (1) defining research questions, (2) searching for relevant studies, (3) applying inclusion and exclusion criteria, and (4) data extraction and synthesis.

Research questions

To guide the systematic mapping and ensure structured data extraction and analysis, three research questions were formulated. These questions reflect both the current state of SRA research and the methodological practices associated with SRA design and evaluation. Table 2 lists the research questions and their goals.

Search strategy

The authors selected the Scopus database for its comprehensive indexing of peer-reviewed literature in software engineering. To complement database retrieval, backward and forward snowballing were applied to ensure the inclusion of studies referenced within the selected papers. Search was limited to English-language, full-text publications after 2010. The search string was iteratively refined through pilot tests: ("software reference architecture" OR "reference architecture" OR "reference model").

Inclusion/exclusion criteria

The following Inclusion/Exclusion criteria were developed to filter out irrelevant papers. After applying these criteria, $n = 75$ primary studies were selected for detailed analysis.

Inclusion criteria (IC)

(1) IC1: The study provides empirical, conceptual, or methodological insights into SRAs.

(2) IC2: The study focuses on the design and evaluation of a Software Reference Architecture.

(3) IC3: The paper contains sufficient information to address the defined research questions

Exclusion criteria (EC)

(1) EC1: Not a primary or empirical study (e.g. review, position paper).

(2) EC2: Not written in English.

(3) EC3: Full text unavailable.

Table 2. List of research questions and how they are addressed

ID	Research questions	Goal
RQ1	What is the current state of research?	To examine publication trends, domains, venues, and collaboration partners to understand the SRA research
RQ2	How are SRA's designed?	To categorize SRA sources of knowledge, design methodologies, frameworks, modeling notations, tools, and architectural artifacts
RQ3	How are SRA's evaluated?	To classify evaluation methods and frameworks, and summarize the quality attributes and metrics used to assess SRAs

Data extraction and synthesis

Data extraction was performed based on the following categories:

(1) Bibliographic Information: Author(s), publication year, venue type (journal/conference), country/region, collaboration type (academic, industrial, or mixed), and Application domain.

(2) SRA Design Characteristics: Knowledge sources (e.g. standards, domain analysis, stakeholder input), design approaches (model-driven, pattern-based, variability-oriented), modeling notations, and design tools.

(3) SRA Evaluation Characteristics: Evaluation methods (e.g. case study, simulation, expert review), metrics and quality attributes (e.g. modifiability, performance, reusability), and measurement methods.

The data extraction process was collaborative to avoid bias. A pilot test was also conducted to ensure the data's relevance. Mixed-method data collection was conducted to provide both qualitative and quantitative feedback. Collected data were stored in a spreadsheet and further analyzed to provide visual presentations.

Results

The findings are structured according to the three research questions. By synthesizing quantitative distributions and qualitative evidence, the results highlight the depth of methodological practices.

Current state of research

SRA research has shown steady and accelerating growth. Early contributions from 2010 to 2014 (12%, $n = 9$) were largely conceptual, introducing foundational frameworks for SRA modeling. However, the period from 2018 to 2025 (33.3%, $n = 25$) witnessed a substantial rise in publications, coinciding with the broader digital transformation in Industry 4.0, cyber-physical systems, and AI-driven infrastructure. The peak output occurred in 2024–2025, reflecting strong interest in SRA across industrial, cloud, and intelligent system domains. This trend signifies a methodological shift from exploratory theoretical studies to empirically validated, domain-oriented SRA's. The analysis reveals nearly one-third of all publications are concentrated in the final two years of the analysis window (2024–2025), strongly suggesting that SRA research is not merely growing but entering an exponential phase of expansion. This trend is likely to continue and even accelerate, driven by several key technological frontiers such as IoT, AI, and Large Language Models (LLM). Figure 1 illustrates the publication trend after 2010.

The analysis of publication venues indicates the field's academic maturation and credibility. A strong majority of the studies (58.7%, $n = 44$) were published in peer-reviewed journals, indicating a focus on disseminating mature, validated, and significant research contributions. Conference papers (41.3%, $n = 31$) serve as a vital platform for presenting novel ideas and early-stage research as well as fostering discussion. Research on the design and evaluation of SRA is disseminated through a diverse set of high-impact, reputable venues. Leading publishers include *Elsevier* ($n = 20$), *Springer* ($n = 13$), *IEEE* ($n = 12$), and *MDPI* ($n = 11$ papers). Prominent journals include Journal of Systems and Software, Software and Systems Modeling, IEEE Access, MDPI Sustainability, Automation in Construction, and Expert Systems. Major conference outlets include ACM, IEEE, and Springer. The publication pattern indicates that the field has reached a significant level of maturity, with research sufficiently well validated for dissemination in peer-reviewed journals.

SRA research has expanded dramatically to address architectural challenges across a wide spectrum of industries. The most prominent category, *Cloud, IoT and Cyber-Physical Systems*

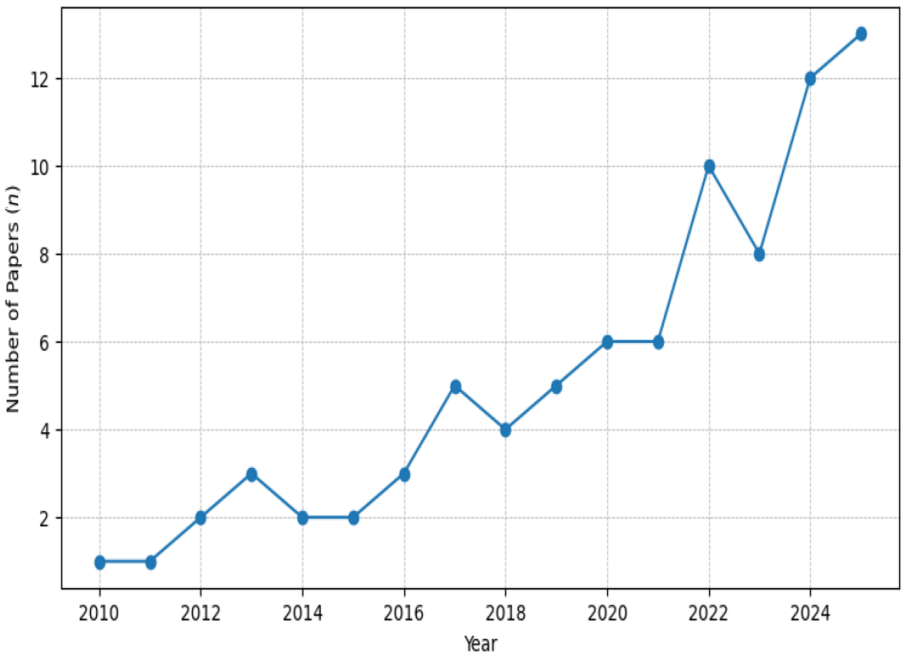

Figure 1. Publication trend of SRA studies after 2010

(24%, $n = 18$), reflects contemporary technological trends and includes SRAs for DevOps in multi-cloud IoT applications, IoT-enabled smart buildings, smart grids, and Industrial IoT. Followed by *Data and Artificial Intelligence* (16%, $n = 12$), encompassing big data systems, cloud-agnostic data analytics, distributed AI services, and LLM-integrated systems. Thirdly, *Healthcare and Well-being* (12%, $n = 9$) includes internet-delivered psychological treatment, virtual reality exposure therapy, health information systems, and healthcare privacy. Moreover, *Manufacturing and Industry 4.0* (14.7%, $n = 11$) includes SRAs for digital manufacturing, digital twin-based predictive maintenance, and logistics. Following this, Security *and Privacy* (14.7%, $n = 11$) focuses on security of cloud systems, IoT privacy, and cybersecurity applications. Furthermore, the *Mobility and Robotics* category (8%, $n = 6$) includes SRAs for autonomous and unmanned vehicles and for social head-gaze generation in social robotics. Finally, research in *Emerging and Niche Domains* (14.7%, $n = 11$) addresses architectures for virtual human integration in the metaverse, blockchain, and culture-aware mobile systems. This diverse domain coverage demonstrates how SRAs have become critical tools for structuring complex systems across virtually all sectors of the modern digital economy. Figure 2 shows the distribution of SRA studies by application domain.

A significant proportion of the studies (49.3%, $n = 37$) were conducted through multi-institutional collaborations, of which international collaborations accounted for ($n = 20$). Furthermore, ($n = 13$) of the studies featured industry-academia collaboration, partnering academic research with industrial practice to ensure relevance and applicability of architecture. Geographically, European institutions lead the research, contributing to ($n = 37$) of the studies, with strong representation from Germany, the Netherlands, and Spain. Seconded by the Asia-Pacific region (22.7%, $n = 17$), led by Australia, followed by America and Brazil (18.7%, $n = 14$). The data suggests that academics largely drive SRA design compared to industries.

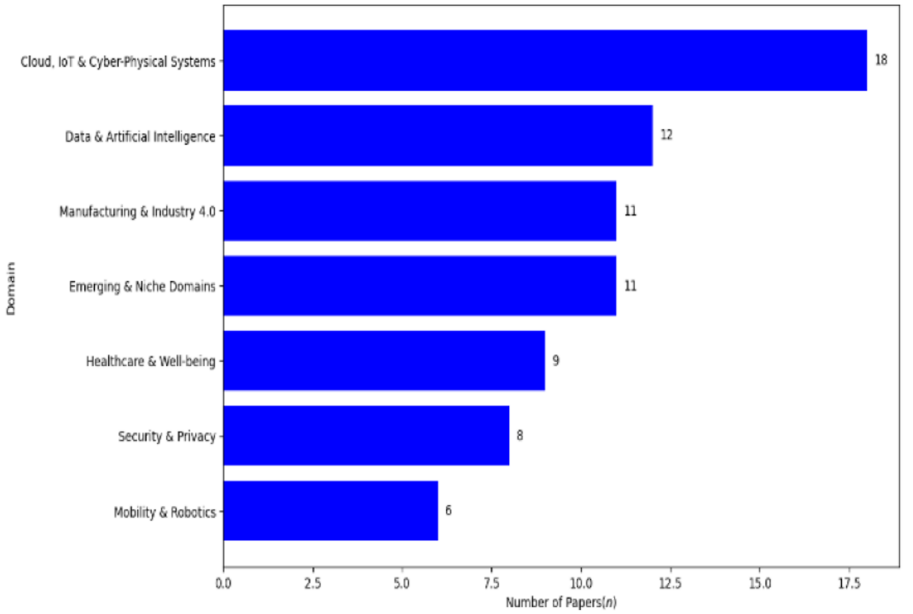

Figure 2. Distribution of SRA studies by application domain

SRA design

Knowledge sources. The design of SRA is informed by four primary knowledge sources, each contributing distinct perspectives and employing specialized methods for knowledge acquisition. The results suggest a combination of sources that have been used. Research shows that studies have utilized multiple sources for knowledge acquisition.

Stakeholder Input (88%, $n = 66$) is the most prevalent method for requirements gathering using semi-structured interviews and structured workshops with architects, developers, and domain experts [27], often supplemented by surveys to validate findings from larger groups [28]. *Domain Analysis* (65.3%, $n = 49$) provides the methodological backbone for systematic design, particularly in product line engineering contexts, by creating structured frameworks and ontology development to establish a precise domain vocabulary [29]. *Formal Standards* (46.7%, $n = 35$) serve as blueprints that assist in designing SRA's through a set of guidelines often tested and verified, such as the ISO/IEC/IEEE 42010 standard for architecture description, which structures documentation and conforms to domain-specific standards [30]. *Existing Architectures* (82.7%, $n = 62$) provide evidence of working models from prior implementations, primarily through systematic studies to learn current designs or patterns, case study analysis, reverse-engineering [31], and understanding evolutionary trends [32]. Table 3 describes four common sources of knowledge identified in this research, together with the methods employed and examples.

Design approach

SRA's are designed using a range of well-established architectural approaches, each contributing unique strengths to the resulting blueprint. Studies have used multiple design approaches to achieve a holistic product. *Pattern-Based Design* (77.3%, $n = 58$) applies recurrent solutions to recurring architectural challenges, drawing on both high-level and fine-grained design patterns. This includes Layered architecture [33, 34], Microservices [30, 35],

Table 3. Sources of knowledge for developing SRA's and gathering methods or tools

Source of knowledge	Methods employed
Stakeholder Input	Semi-structured Interviews
	Structured Surveys and Questionnaires
	Focus Groups
Domain Analysis	Feature Modelling
	Scope, Commonalities, and Variability (SCV) Analysis
	Ontology Development
	Domain-Specific Language (DSL) Design
Formal Standards	Compliance Gap Analysis
	Adherence to Standardized Metamodels (e.g. ISO/IEC/IEEE 42010)
	Implementation of Protocol and Interface Specifications (e.g. HL7 FHIR, MQTT)
Existing Architectures	Systematic Reviews/Mapping Studies
	Reverse-engineering
	Mining Software Repositories
	Comparative Analysis of Multiple Architectures

pipes-and-filters [36], and MAPE-K. Complementing this, *Model-Driven Engineering (MDE)* (69.3%, $n = 52$) relies on abstraction through formal or semi-formal models to manage architectural complexity. Modeling languages such as UML [37, 38], SysML [32], and emerging ArchiMate [39–41] capture structural and behavioral details. *Component-Based Design* (66.7%, $n = 50$) is another approach, which decomposes systems into cohesive, loosely coupled components with well-defined interfaces. This technique enhances critical quality attributes such as reusability, maintainability, and replaceability. It is applied to multi-agent microgrid management [42], decentralized warehouse control [43], and self-adaptive middleware for wireless sensor networks [44].

Managing architectural complexity is further supported by the *Viewpoint-Based Approach* (65.3%, $n = 49$), which structures architectural descriptions into coherent views aligned with stakeholders. Guided by ISO/IEC/IEEE 42010 frameworks such as the 4 + 1 View Model, Views and Beyond, and the RM-ODP reference model [36, 45–49]. *Methodology-Driven Design* (54.7%, $n = 41$), in which Design Science Research (DSR) methodology is prevalent, develops artifacts through iterative cycles of problem identification, construction, demonstration, and evaluation [27, 50, 51]. Dedicated SRA methodologies, such as ProSA-RA [52, 53], and empirically grounded multi-phase methods reinforce rigor, traceability, and repeatability [54–56].

Domain-Driven Design (DDD) (36%, $n = 27$) strengthens SRA development by embedding domain semantics, as supported by Feature-Oriented Domain Analysis (FODA) to model commonalities and variabilities in domains such as smart farming [47] and digital-twin predictive maintenance [28]. Finally, *Standards-Based Design* (34.7%, $n = 26$) is critical in regulated environments, incorporating international standards and technical specifications, including IEC 62264, the Industrial Internet Reference Architecture (IIRA), and NIST cybersecurity guidelines to ensure compliance with industry's best practices [57–59]. Figure 3 shows the different approaches used for designing SRAs.

SRA design is substantiated by three main categories of *supporting frameworks* and methods that provide structured guidance for implementation. They are *Architectural frameworks and standards* (46.7%, $n = 35$) [36, 39, 58–62], *Development methodologies* (54.7%, $n = 41$) [27, 50–52, 54–57, 63], and *Domain-specific standards and reference models* (34.7%, $n = 26$) [42, 57, 59, 64]. Moreover, several modeling notations were employed to articulate architectural structures, behaviors, and domain concepts. Table 4 summarizes the various modeling notations used in the design of SRA, as well as the specific categories and their frequencies within each category.

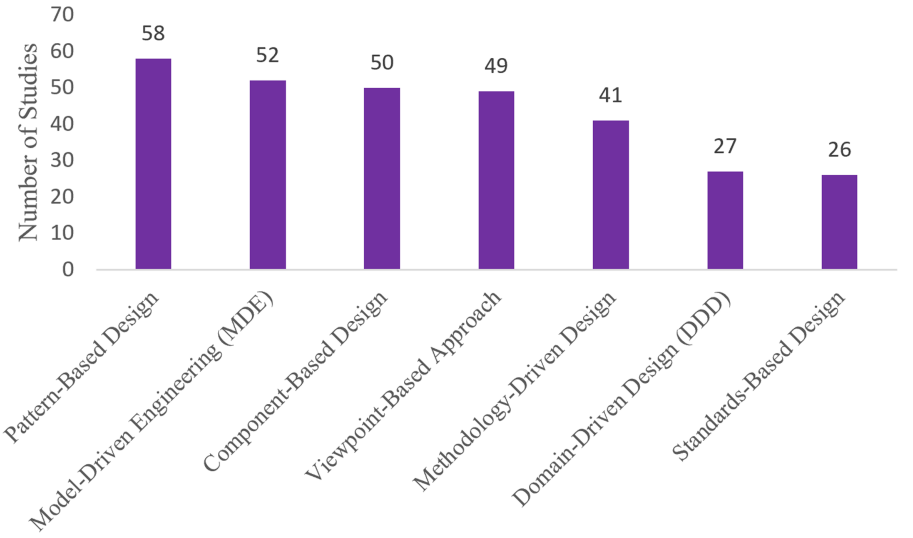

Figure 3. Frequency distribution by design approach

Table 4. A summary of the modeling notations category, together with specific notations and frequency

Modeling notation category	Specific notations	Frequency
Informal Diagrams	Box-And-Line, Conceptual Diagrams	70
UML Family	Class, Component, Deployment Diagrams	57
Domain-Specific	Feature Models, BPMN, DSLs	25
ArchiMate	Multi-Layer Architecture Modeling	20
Ontology Languages	Rdf/Owl, Sparql	18
Formal ADLs	Pi-ADL, Montiarc, Sysml	12

SRA evaluation

Evaluation methods

Multiple evaluation methods were identified for assessing SRA's, with many studies employing a multi-method approach to strengthen validation and demonstrate comprehensive evaluation. *Case Study* evaluation was the most prevalent approach (70.7%, $n = 53$), typically involving instantiating the SRA into a prototype or full system within a specific domain context to demonstrate feasibility and utility [38, 42]. *Expert review* (30.7%, $n = 23$) was conducted through structured workshops, interviews, or surveys to identify design flaws and risks, such as using the Architecture Tradeoff Analysis Method (ATAM) workshops and the Framework for Evaluation of Reference Architectures (FERA) [52, 65]. *Prototype implementation* (41.3%, $n = 31$) served as concrete technical validation. *Comparative analysis* (16%, $n = 12$) involved systematic comparison against existing alternatives [66], while *simulation* (5.3%, $n = 4$) was reserved for domains where real deployment was impractical, such as in unmanned forest management systems [67].

The rigorous evaluation of an SRA necessitates *structured methodologies* to assess its quality, utility, and robustness. Beyond selecting evaluation types such as case studies or expert reviews, researchers frequently employ established architectural frameworks to guide comprehensive analysis. Namely,

Architecture Tradeoff Analysis Method (ATAM) (10.7%, $n = 8$) was utilized to identify risks and tradeoffs among quality attributes like modifiability and security [38, 65, 68,

69], *Software Architecture Analysis Method (SAAM)* (9.3%, $n = 7$) was applied to assess modifiability and extensibility through change impact analysis [30, 70], *Framework for Evaluation of Reference Architectures (FERA)* (6.7%, $n = 5$) provided a dedicated checklist-based approach for evaluating documentation completeness and clarity [44, 52, 57].

Beyond these specialized frameworks, broader evaluation paradigms were also employed, including *Design Science Research (DSR) principles* that assessed feasibility and usefulness through case studies and surveys, and *Standardized Questionnaires* such as System Usability Scale (SUS), Usability Evaluation Questionnaire (UEQ), and Technology Adoption Model (TAM) that provided quantitative measures of usability and acceptance [27, 29, 41, 51]. Figure 4 illustrates the frequency of different evaluation methods.

Evaluation metrics

Development and Maintenance Qualities emerge as the most extensively evaluated category, with reusability being the most frequently assessed attribute (74.7%, $n = 56$), typically measured through code reuse ratios and development effort metrics [45, 71]. Extensibility is another prominent concern (60%, $n = 45$), evaluated through component integration ease [38], while maintainability (53.3%, $n = 40$) and modifiability (52%, $n = 39$) are analyzed via code metrics and scenario-based methods [30]. *Operational Qualities* constitute the second major category, with performance evaluation appearing in ($n = 54$) studies through throughput and latency metrics, scalability ($n = 48$) via load testing, and resource efficiency in ($n = 35$), focusing on energy and computational consumption.

Functional and Business Qualities demonstrate substantial presence, with usability evaluation (49.3%, $n = 37$) using standardized scores, functional correctness (44%, $n = 33$) through requirement coverage, and completeness in (36%, $n = 27$) against domain needs [27, 39, 57]. *Cross-Cutting Concerns* show significant variation, with security appearing most frequently (56%, $n = 42$) through threat coverage analysis, interoperability (45.3%, $n = 34$), and adaptability (41.3%, $n = 31$) through change response evaluation [36, 62, 72]. Table 5 of shows the detailed categories, quality, and attributes.

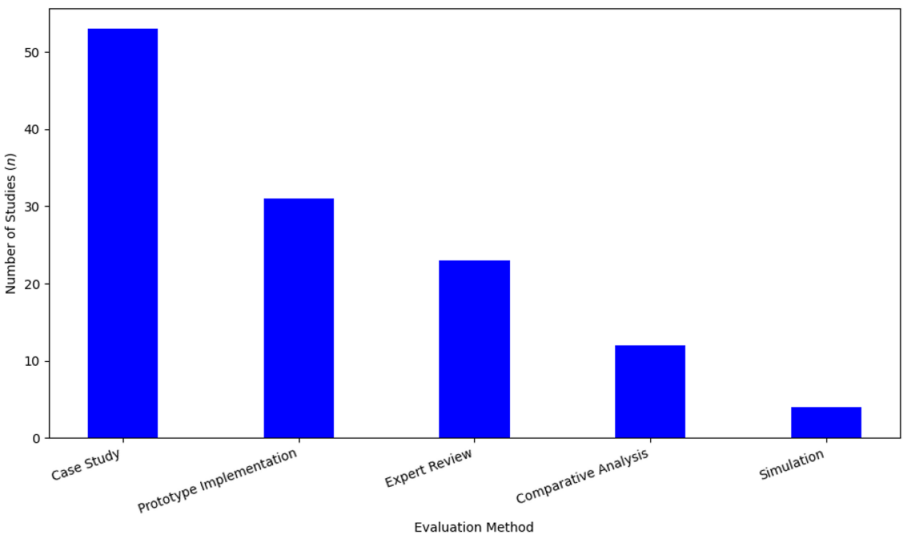

Figure 4. Shows evaluation methods and their frequency as employed by studies

Table 5. The table shows the detailed category, its quality and attributes used as quality metrics

Category	Quality	Attribute
Development and Maintenance Qualities	Scalability	Load testing, capacity analysis
	Extensibility	Component integration ease, scenario analysis
	Maintainability	Code metrics, modification effort analysis
	Modifiability	Scenario-based methods, architectural analysis
Operational Qualities	Performance	Throughput, latency, response time metrics
	Scalability	Load testing, capacity analysis
	Resource Efficiency	Energy consumption, computational resources
	Reliability	Uptime analysis, fault tolerance testing
Functional and Business Qualities	Usability	Standardized scores, user feedback
	Functional Correctness	Requirement coverage, output verification
	Completeness	Domain coverage, stakeholder requirements
	Applicability	Domain relevance, implementation feasibility
Cross-Cutting Concerns	Security	Threat coverage, compliance checks
	Interoperability	Integration capabilities, standard conformance
	Adaptability	Change response evaluation, scenario analysis
	Privacy	Data protection mechanisms, regulatory compliance

Discussion

Findings

SRA research has expanded rapidly over the past decade, shifting from predominantly conceptual proposals to empirically grounded, domain-oriented studies, with a marked surge in 2018–2025 and peak activity in 2024–2025. Publication venues indicate the field's maturation; most contributions appear in peer-reviewed journals, while conferences remain important for early dissemination. Domain distribution mirrors global digitalization trends with Cloud/IoT/CPS, Data and AI, and Industry 4.0 accounting for the largest share, followed by healthcare and well-being, security and privacy, mobility and robotics, and several other niche domains. Collaboration patterns show a highly networked community; nearly half of all papers are multi-institutional, and a substantial proportion are international. Industry-academia collaboration is comparatively limited, offering an opportunity to narrow the research-practice gap.

SRA design is consistently multi-source. Across studies, four main sources of architectural knowledge are identified: stakeholder input, followed by analysis of existing architectures, domain analysis, and formal standards, and most designs combine more than one. The selection method is strongly domain-dependent. This underlines that knowledge sourcing reflects regulatory intensity, system criticality, and technological maturity rather than a single best practice. SRA design relies on a combination of complementary approaches rather than a single method. Pattern-based, model-driven, component-based, and viewpoint-based techniques form the core design foundation, supported by formal methodologies such as design science research and domain-specific or standard-driven practices. Designers increasingly use multi-notation modeling that combines UML, SysML, ArchiMate, feature models, and informal diagrams to capture different architectural perspectives. A wide range of artifacts, from reference models and meta-models to component libraries, workflow diagrams, and prototypes, demonstrates that SRA design is both systematic and practice-oriented, integrating conceptual rigor with real-world applicability.

Evaluation practice is characterized by triangulation. Mostly, studies combined case studies with prototype implementation, expert review, comparative analyses, and controlled experiments, while simulations are the least common but have gained demand in recent years. Established frameworks (ATAM, SAAM, FERA) provide structure, while controlled user/field experiments remain comparatively rare. Measured qualities include development and operation, reusability, maintainability, modifiability, and performance and scalability occur most frequently, but there is no universal metric set. Measurement choices are consistently tailored to domain objectives and constraints.

Implications

The implications outlined below are intended to guide researchers working to advance the theoretical and methodological foundations of SRA design and evaluation.

(1) Increased collaboration between industry and academia will enable SRAs to address real operational constraints and deployment environments. Such partnerships will also enhance external validity and facilitate practical adoption.

(2) The widespread use of specialized frameworks such as ProSA-RA, FERA, etc., suggests that SRA development benefits from formalized methodologies. Researchers and practitioners should adopt these structured processes to enhance the reproducibility and quality of the resulting architecture.

(3) Most studies use multiple modeling notations, including UML, SysML, ArchiMate, and domain-specific diagrams, suggesting designers embrace a multi-notation toolkit to capture richer architectural communication.

(4) Develop standardized evaluation protocols and metric baselines. Current evaluations are diverse and hard to compare. Establishing shared metric sets, benchmark tasks, and reusable evaluation frameworks would improve consistency and reproducibility across studies.

(5) Tool support is critical for operational adoption. Automated model checking, configuration assistance, and code generation pipelines can make SRAs more maintainable and easier to instantiate consistently.

(6) Build shared repositories of architectural models and reusable assets. Open access component libraries, model templates, and instantiation examples would accelerate reuse and help establish baselines for comparison and extension.

(7) Investigate SRA challenges in fast-evolving digital ecosystems. Domains involving AI-driven autonomy, distributed edge computing, and digital twins introduce dynamic architectural demands. Research is needed to understand how SRAs can evolve and adapt in these environments.

Threats to validity

This study may be subject to a few threats to validity. This study primarily relied on the Scopus database for its comprehensive coverage of peer-reviewed software engineering research and its robust indexing of major publishers. However, studies indexed exclusively in other digital libraries (e.g., IEEE Xplore, ACM Digital Library, Web of Science) may not have been captured. The process of determining study eligibility required interpretation of titles, abstracts, and, in some cases, full texts. Judgments were particularly necessary when studies discussed architectural frameworks without explicitly labeling them as SRAs, and extracting design and evaluation characteristics required interpreting narrative descriptions and architectural documentation. Variation in terminology, differing levels of reporting detail, and heterogeneous documentation formats posed challenges for consistent extraction.

Conclusion

This systematic review synthesized findings from $n = 75$ primary studies published between 2010 and 2025, offering a comprehensive overview of how SRA is designed and evaluated. The results show that SRA design depends on synthesizing multiple sources of knowledge, including stakeholder needs, existing systems, domain analysis, and formal standards applied in proportions that reflect each domain's requirements. A core set of design approaches emerged as consistently dominant, such as pattern-based, model-driven engineering, component and modular design, and viewpoint-based modeling. The review also indicates that evaluation practices are becoming increasingly structured and evidence-based. While case studies and prototype implementations remain foundational, expert review and selective quantitative assessment are increasingly incorporated to strengthen evaluation. As software ecosystems continue to expand and integrate emerging technologies such as AI, digital twins, and distributed autonomy, the role of SRAs as reusable architectural blueprints will become even more critical. The findings of this systematic review highlight several promising directions for advancing research and practice in SRAs. Advancing methodological rigor, practical relevance, and tool maturity will therefore be essential to enhancing the long-term effectiveness and adoption of SRAs.

Author contributions

All authors' contributed equally to this article.

Data availability statement

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Cervantes H, Kazman R. Designing software architectures: a practical approach. Addison-Wesley Professional; 2024.
2. Kusmin M, Laanpere M. Development and validation of a reference architecture for the smart schoolhouse. Balt J Mod Comput. 2025; 13(2). doi: 10.22364/bjmc.2025.13.2.02.
3. Söylemez M, Tekinerdogan B, Tarhan AK. Microservice reference architecture design: a multi-case study. Softw Pract Exp. 2024; 54(1): 58-84. doi: 10.1002/spe.3241.
4. Serôdio C, Mestre P, Cabral J, Gomes M, Branco F. Software and architecture orchestration for process control in Industry 4.0 enabled by cyberphysical systems technologies. Appl Sci. 2024; 14 (5): 2160. doi: 10.3390/app14052160.
5. Ataei P, Litchfield A. The state of big data reference architectures: a systematic literature review. IEEE Access. 2022; 10: 113789-807. doi: 10.1109/access.2022.3217557.
6. Zhang M, Teng Y, Kong H, Baugh J, Su Y, Mi J, Du B. Automatic modelling and verification of Autosar architectures. J Syst Softw. 2023; 201: 111675. doi: 10.1016/j.jss.2023.111675.
7. Martínez-Fernández S, Ayala CP, Franch X, Marques HM. Benefits and drawbacks of software reference architectures: a case study. Inf Softw Technol. 2017; 88: 37-52. doi: 10.1016/j.infsof.2017.03.011.
8. Petersen K, Vakkalanka S, Kuzniarz L. Guidelines for conducting systematic mapping studies in software engineering: an update. Inf Softw Technol. 2015; 64: 1-18. doi: 10.1016/j.infsof.2015.03.007.
9. Petersen K, Feldt R, Mujtaba S, Mattsson M Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE); 2008.
10. Bass L, Clements P, Kazman R. Software architecture in practice. 4th ed. Addison-Wesley Professional; 2021.

11. Rozanski N, Woods E. Software systems architecture: working with stakeholders using viewpoints and perspectives. 2nd ed. Wiley; 2012.
12. Souza VJ, Neves VO, Kimura BY. Dependable microservices in the kubernetes era: a practitioners survey. J Internet Serv Appl. 2024; 15(1): 561-83.
13. Söylemez M, Tekinerdogan B, Kolukısa Tarhan A. Feature-driven characterization of microservice architectures: a survey of the state of the practice. Appl Sci. 2022; 12(9): 4424. doi: 10.3390/app12094424.
14. Nakagawa EY, Antonino PO. Reference architectures for critical domains. Cham: Springer; 2023.
15. Valle PHD, Garcés L, Volpato T, Martínez-Fernández S, Nakagawa EY. Towards suitable description of reference architectures. PeerJ Comput Sci. 2021; 7: e392. doi: 10.7717/peerj-cs.392.
16. Galster M. Software reference architectures: related architectural concepts and challenges. In: Proceedings of the 1st International Workshop on Exploring Component-Based Techniques for Constructing Reference Architectures; 2015.
17. Santana EFZ, Chaves AP, Gerosa MA, Kon F, Milojicic DS. Software platforms for smart cities: concepts, requirements, challenges, and a unified reference architecture. ACM Comput Surv (Csur). 2017; 50(6): 1-37. doi: 10.1145/3124391.
18. Martínez-Fernández S, Ayala CP, Franch X, Marques HM. Benefits and drawbacks of software reference architectures: a case study. Inf Softw Technol. 2017; 88: 37-52. doi: 10.1016/j.infsof.2017.03.011.
19. Bucaioni A, Di Salle A, Iovino L, Malavolta I, Pelliccione P. Reference architectures modelling and compliance checking. Softw Syst Model. 2023; 22(3): 891-917. doi: 10.1007/s10270-022-01022-z.
20. Martínez-Fernández S, Ayala CP, Franch X, Marques HM, Ameller D. Towards guidelines for building a business case and gathering evidence of software reference architectures in industry. J Softw Eng Res Dev. 2014; 2(1): 7. doi: 10.1186/s40411-014-0007-5.
21. Muller G. A reference architecture primer. Eindhoven Univ. of Techn., Eindhoven, White Paper: 24, 2008.
22. Sittón-Candanedo I, Alonso RS, Corchado JM, Rodríguez-González S, Casado-Vara R. A review of edge computing reference architectures and a new global edge proposal. Future Generation Comput Syst. 2019; 99: 278-94. doi: 10.1016/j.future.2019.04.016.
23. Abtoy A, Touhafi A, Tahiri A. Ambient Assisted living system's models and architectures: a survey of the state of the art. J King Saud Univ-Comput Inf Sci. 2020; 32(1): 1-10.
24. Savold R, Dagher N, Frazier P, Mccallam D Architecting cyber defense: a survey of the leading cyber reference architectures and frameworks. In: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud); 2017.
25. Garcés L, Martínez-Fernández S, Oliveira L, Valle P, Ayala C, Franch X, Nakagawa EY. Three decades of software reference architectures: a systematic mapping study. J Syst Softw. 2021; 179: 111004. doi: 10.1016/j.jss.2021.111004.
26. Muccini H, Vaidhyanathan K. Software architecture for ml-based systems: what exists and what lies ahead. In: 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN); 2021.
27. Weber T, Buchkremer R. Blockchain-based reference architecture for automated, transparent, and notarized attestation of compliance adaptations. Appl Sci. 2022; 12(9): 4531. doi: 10.3390/app12094531.
28. Van Dinter R, Tekinerdogan B, Catal C. Reference architecture for digital twin-based predictive maintenance systems. Comput Ind Eng. 2023; 177: 109099. doi: 10.1016/j.cie.2023.109099.
29. Ortúzar F, Rossel PO, Gutierrez FJ. A reference architecture to support the collaborative development of virtual reality exposure therapy software. In: 2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD); 2024.

30. Mukhiya SK, Lamo Y, Rabbi F. A reference architecture for data-driven and adaptive internet-delivered psychological treatment systems: software architecture development and validation study. JMIR Hum Factors. 2022; 9(2): e31029. doi: 10.2196/31029.

31. Martnez-Fernndez S, Ayala CP, Franch X, Marques HM. Benefits and drawbacks of software reference architectures. Inf Softw Technol. 2017; 88(C): 37-52.

32. Enos ES, Herber DR. A reference architecture for on-premises software-defined infrastructure using model-based systems engineering. IEEE Access. 2025.

33. Candela L, Castelli D, Pagano P. A reference architecture for digital library systems. In: DELOS Conference on Digital Libraries. Grand Hotel Continental-Tirrenia, Pisa; 2007.

34. Lewis G, Novakouski M, Sánchez E. A reference architecture for group-context-aware mobile applications. In: International Conference on Mobile Computing, Applications, and Services; 2012.

35. Wang Y, Li S, Liu H, Zhang H, Pan B. A reference architecture for blockchain-based traceability systems using domain-driven design and microservices. In: 2022 29th Asia-Pacific Software Engineering Conference (APSEC); 2022. p. 269-78. doi: 10.1109/APSEC57359.2022.00039.

36. Hildebrandt D. A software reference architecture for service-oriented 3D geovisualization systems. ISPRS Int J Geo-Inf. 2014; 3(4): 1445-90. doi: 10.3390/ijgi3041445.

37. Schroeder J, Holzner D, Berger C, Hoel C-J, Laine L, Magnusson A. Design and evaluation of a customizable multi-domain reference architecture on top of product lines of self-driving heavy vehicles—an industrial case study. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE). 2015; 2: 189-98. doi: 10.1109/ICSE.2015.147.

38. Wang Y, Li S, Liu H, Zhang H, Pan B. A reference architecture for blockchain-based traceability systems using domain-driven design and microservices. In: 2022 29th Asia-Pacific Software Engineering Conference (APSEC); 2022. p. 269-78. doi: 10.1109/APSEC57359.2022.00039.

39. Iacob M, van Sinderen MJ, Steenwijk M, Verkroost P. Towards a reference architecture for fuel-based carbon management systems in the logistics industry. Inf Syst Front. 2013; 15(5): 725-45. doi: 10.1007/s10796-013-9416-y.

40. Pääkkönen P, Horsmanheimo S, Pakkala D, Tuomimäki L, Backman J. Reference architecture design and evaluation for digitalization of underground mining. Internet of Things. 2024; 26: 101238. doi: 10.1016/j.iot.2024.101238.

41. Simons R, Eshuis R, Ozkan B. A reference architecture for reverse logistics in the high-tech industry. Comput Ind Eng. 2024; 194: 110368. doi: 10.1016/j.cie.2024.110368.

42. Garcia-Rodriguez S, Sleiman HA, Nguyen V. A multi-agent system architecture for microgrid management. Anonymous. 2016: 55-67. doi: 10.1007/978-3-319-40159-1_5.

43. Verriet J, van Wijngaarden B. A reference architecture capturing structure and behaviour of warehouse control. In: Automation in Warehouse Development; 2011.

44. Portocarrero JM, Delicato FC, Pires PF, Costa B, Li W, Si W, Zomaya AY. RAMSES: a new reference architecture for self-adaptive middleware in wireless sensor networks. Ad Hoc Networks. 2017; 55: 3-27. doi: 10.1016/j.adhoc.2016.11.004.

45. Ozkaya M, Turunc A. A reference architecture for smart car parking management systems. Syst. 2025; 13(2): 70. doi: 10.3390/systems13020070.

46. Male G, Shaghouei E, Pattinson C. Towards the development of a culture aware reference architecture for mobile technology: a tools and methodology perspective. In: 2017 IEEE Africon; 2017.

47. Krisnawijaya NNK, Tekinerdogan B, Catal C, van der Tol R. Reference architecture design for developing data management systems in smart farming. Ecol Inform. 2024; 81: 102613. doi: 10.1016/j.ecoinf.2024.102613.

48. Tertulino R, Ivaki N, Morais H. Design a software reference architecture to enhance privacy and security in electronic health records. IEEE Access. 2024.

49. Tummers J, Tobi H, Catal C, Tekinerdogan B. Designing a reference architecture for health information systems. BMC Med Inform Decis Making. 2021; 21: 1-14. doi: 10.1186/s12911-021-01570-2.
50. Bashir MR, Gill AQ, Beydoun G. A reference architecture for IoT-enabled smart buildings. SN Comput Sci. 2022; 3(6): 493. doi: 10.1007/s42979-022-01401-9.
51. Kratsch W, König F, Röglinger M. Shedding light on blind spots–developing a reference architecture to leverage video data for process mining. Decis Support Syst. 2022; 158: 113794. doi: 10.1016/j.dss.2022.113794.
52. Nakagawa EY, Oquendo F, Maldonado JC. Reference architectures. Softw Architecture. 2014; 1: 55-82.
53. Oliveira LBR, Nakagawa EY. A service-oriented reference architecture for software testing tools. In: European Conference on Software Architecture; 2011.
54. Ataei P, Litchfield AT. Towards a domain-driven distributed reference architecture for big data systems. In: AMCIS; 2023.
55. Nadal S, Herrero V, Romero O, Abelló A, Franch X, Vansummeren S, Valerio D. A software reference architecture for semantic-aware big data systems. Inf Softw Technol. 2017; 90: 75-92. doi: 10.1016/j.infsof.2017.06.001.
56. Tran NK, Pallewatta S, Babar MA. An empirically grounded reference architecture for software supply chain metadata management. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering; 2024.
57. Freire G, Vasconcelos A. RAMOM: a reference architecture for manufacturing operations management activities in Industry 4.0. In: Proceedings of the 26th International Conference on Enterprise Information Systems; 2024.
58. da Rocha H, Abrishambaf R, Pereira J, Espirito Santo A. Integrating the IEEE 1451 and IEC 61499 standards with the industrial internet reference architecture. Sensors. 2022; 22(4): 1495. doi: 10.3390/s22041495.
59. Kampourakis V, Gkioulos V, Katsikas S. A step-by-step definition of a reference architecture for cyber ranges. J Inf Security Appl. 2025; 88: 103917. doi: 10.1016/j.jisa.2024.103917.
60. Bucaioni A, Weyssow M, He J, Lyu Y, Lo D. A functional software reference architecture for LLM-integrated systems. In: 2025 IEEE 22nd International Conference on Software Architecture Companion (ICSA-C), Odense, Denmark, Mar. 31–Apr. 4; 2025. p. 1-5. doi: 10.1109/ICSA-C65153.2025.00006.
61. Mateo-Casali MA, Boza A, Fraile F, Moya-Ruiz L. Reference architecture for digital product passports: leveraging AI in Industry 4.0 systems. 2025 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Jun. 16–19; 2025. p. 1-9. doi: 10.1109/ICE/ITMC65658.2025.11106524.
62. Wehrmeister K, Pastor A, Carreras Rodriguez L, Monti A. Big data reference architecture for the energy sector. Sustainability. 2025; 17(14): 6488. doi: 10.3390/su17146488.
63. Nakagawa EY, Oliveira Antonino P, Becker M. Reference architecture and product line architecture: a subtle but critical difference. In: European Conference on Software Architecture; 2011.
64. Itäpelto T, Elhajj M, van Sinderen M, Iacob M. Reference architecture of cybersecurity digital twin. In: International Conference on Enterprise Design, Operations, and Computing (EDOC) 2024 Workshops, Revised Selected Papers. Cham: Springer; 2025. p. 389-404. doi: 10.1007/978-3-031-79059-1_24.
65. Norta A, Grefen P, Narendra NC. A reference architecture for managing dynamic inter-organizational business processes. Data Knowl Eng. 2014; 91: 52-89. doi: 10.1016/j.datak.2014.04.001.
66. Janbi N, Mehmood R, Katib I, Albeshri A, Corchado JM, Yigitcanlar T. Imtidad: a reference architecture and a case study on developing distributed AI services for skin disease diagnosis over cloud, fog and edge. Sensors. 2022; 22(5): 1854. doi: 10.3390/s22051854.

67. Park S, Park YB. ITE arbitrator: a reference architecture framework for sustainable IT ecosystems. In: Proceedings of the 4th International Workshop on Software Engineering for Systems-of-Systems; 2016.

68. Chauhan MA, Babar MA, Sheng QZ. A reference architecture for provisioning of tools as a service: meta-model, ontologies and design elements. Future Generation Comput Syst. 2017; 69: 41-65. doi: 10.1016/j.future.2016.12.002.

69. Ataei P. A reference architecture for large scale distributed domain-driven big data systems, Ph.D. dissertation, Auckland University of Technology, Auckland, New Zealand; 2024.

70. Srinivasan V, Murphy RR, Bethel CL. A reference architecture for social head gaze generation in social robotics. Int J Soc Robot. 2015; 7(5): 601-16. doi: 10.1007/s12369-015-0315-x.

71. Ignaim K, Fernandes JM, Ferreira AL. An industrial experience of using reference architectures for mapping features to code. Sci Comput Program. 2024; 234: 103087. doi: 10.1016/j.scico.2024.103087.

72. Affonso FJ, Nakagawa EY. A reference architecture based on reflection for self-adaptive software. In: 2013 VII Brazilian Symposium on Software Components, Architectures and Reuse; 2013.

Further reading

73. Xu J, Randell B, Romanovsky A. A generic approach to structuring and implementing complex fault-tolerant software. In: Proceedings of the Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2002), Washington, DC, USA; 2002. p. 207-214. doi: 10.1109/ISORC.2002.1003680.

74. Latha DS, Samanchuen T. Development of reference process model and reference architecture for pharmaceutical cold chain. Sustainability. 2023; 15(5): 3935. doi: 10.3390/su15053935.

75. Neureiter C, Uslar M, Engel D, Lastro G. A standards-based approach for domain specific modelling of smart grid system architectures. In: 2016 11th System of Systems Engineering Conference (SoSE), Kongsberg, Norway; 2016. p. 1-6. doi: 10.1109/SYSOSE.2016.7542888.

76. Spyrou O, Hurst W, Krampe C. A reference architecture for virtual human integration in the metaverse: enhancing the galleries, libraries, archives, and museums (GLAM) sector with AI-driven experiences. Future Internet. 2025; 17(1): 36. doi: 10.3390/fi17010036.

77. Splettstößer A, Ellwein C, Wortmann A. Self-adaptive digital twin reference architecture to improve process quality. Proced CIRP. 2023; 119: 867-72.

Corresponding author

Sailesh Saras Chand can be contacted at: sailesh.chand@fnu.ac.fj