Frontiers in Emerging Computer Science and Information Technology (Open Access)

Volume 02, Issue 02, February 2025,
Publish Date: 01-02-2025
PageNo.01-07

Architectural Patterns and Strategies for Software Interoperability: A Comprehensive Catalog Approach

Dr. Fatima Zahra El Amrani
Department of Computer Science, Mohammed V University, Rabat, Morocco

ABSTRACT

Achieving seamless interoperability among diverse software systems remains a central challenge in modern enterprise and distributed computing environments. This study presents a comprehensive catalog of architectural patterns and strategies specifically designed to address software interoperability. The catalog synthesizes established and emerging approaches, including service-oriented architecture (SOA), microservices, messaging middleware, canonical data models, API gateways, and event-driven designs. Each pattern is systematically analyzed in terms of context, applicability, advantages, and trade-offs. Additionally, the work proposes a classification framework to guide architects in selecting appropriate strategies based on system requirements, integration complexity, and scalability considerations. The catalog aims to serve as both a reference and a decision-support tool for practitioners seeking to enhance interoperability while maintaining flexibility, maintainability, and performance across heterogeneous software ecosystems.

KEYWORDS: Software interoperability, architectural patterns, integration strategies, service-oriented architecture, microservices, messaging middleware, API gateways, event-driven architecture, canonical data model, software architecture catalog.

INTRODUCTION

In today's interconnected digital landscape, software systems rarely operate in isolation. The ability of disparate systems to exchange and use information effectively, known as interoperability, has become a paramount concern for organizations across various domains, including healthcare [Benson and Grieve 2016; Daliya and Ramesh 2019; Gazzarata et al. 2017; Guo et al. 2011; Garcés 2018], smart homes [Farooq et al. 2020], and industrial automation [Burns et al. 2019]. The lack of interoperability can lead to significant challenges, including data silos, inefficient workflows, increased operational costs, and missed opportunities for collaboration and innovation [Abukwaik and Rombach 2017; Maciel et al. 2017]. As systems become more complex and interconnected, often forming "Systems of Systems" (SoS) [Madni and Sievers 2014; Baldwin et al. 2017; Ingram et al. 2015], ensuring seamless interaction becomes an architectural imperative [Bouziat et al. 2018].

Interoperability is a multi-faceted concept, encompassing various layers from technical connectivity to semantic understanding and organizational alignment [Kubicek et al. 2011; eHGI 2017; Aydin and Aydin 2020]. Achieving it requires careful architectural design and the application of proven strategies. Software architecture, as defined by Bass [2013] and Garlan [2007], provides the fundamental structures of a software system and the discipline for creating such structures. It dictates how components interact and thus plays a crucial role in enabling or hindering interoperability. While various architectural patterns and solutions exist to address interoperability challenges [Keshav and Gamble 1998; Spalazzese and Inverardi 2010; Valle et al. 2019], there is a recognized need for a structured and comprehensive catalog that systematizes these solutions. Such a catalog could serve as a valuable resource for architects, developers, and decision-makers in designing, evaluating, and implementing interoperable software systems.

Existing research has explored interoperability frameworks [Ibrahim and bin Hassan 2010; Muketha et al. 2014; Farooq et al. 2020], architectural strategies [Valle et al. 2019; Valle et al. 2021], and even taxonomies of interoperability types [Maciel et al. 2024; Noura et al. 2019]. However, a consolidated, detailed catalog specifically focusing on architectural *solutions* – the concrete patterns and approaches that facilitate interoperability across different layers – is still evolving. This article proposes the conceptual framework for such a catalog, outlining its structure, the types of solutions it would encompass, and its potential benefits. By providing a systematic organization of architectural knowledge pertaining to interoperability, this catalog aims to enhance the design process, improve

reusability, and reduce the inherent complexities of integrating diverse software systems.

METHODS

Developing a comprehensive catalog of interoperability architectural solutions for software systems requires a rigorous and systematic approach. The hypothetical methodology outlined below draws upon established research methods in software engineering and aims to ensure the catalog's completeness, accuracy, and practical utility.

Definition of Interoperability and Its Layers

Before identifying solutions, a clear and consistent definition of interoperability and its various layers is essential. Drawing upon established standards and literature [IEEE 2000; Kubicek et al. 2011; eHGI 2017], interoperability can be broadly categorized into:

- Technical Interoperability: Concerns the protocols and interfaces that allow systems to exchange data at a low level [van der Veer and Wiles 2008]. This includes network protocols, data formats, and communication mechanisms.
- Syntactic Interoperability: Focuses on the structure of data exchange, ensuring that systems can parse and understand the format of the information received [Aydin and Aydin 2020; Bicer et al. 2005]. This typically involves agreed-upon message structures (e.g., XML, JSON, HL7).
- Semantic Interoperability: The most challenging layer, ensuring that the meaning of the exchanged information is unambiguously understood by all participating systems [Moreira et al. 2018; Rahman and Hussain 2020]. This often requires shared ontologies, controlled vocabularies, and semantic mapping.
- Organizational/Pragmatic Interoperability: Relates to the ability of organizations and people to cooperate towards mutually beneficial goals through shared business processes and policies [Adamo et al. 2018; Benany and Beqqali 2018].

A typology of architectural strategies for interoperability, such as TASIS [Valle et al. 2025] or the one proposed by Valle et al. [2019], would guide the categorization of solutions within these layers.

Data Collection and Identification of Architectural Solutions

The identification of architectural solutions would involve a multi-pronged data collection strategy:

1. Systematic Literature Review (SLR): A comprehensive SLR would be conducted across major scientific databases (e.g., ACM Digital Library, IEEE Xplore, Scopus, Web of Science) using keywords related to "software architecture," "interoperability," "patterns," "strategies," "integration," "mediators," "reference architectures," and specific interoperability layers. This would help identify existing research on architectural solutions for interoperability [Valle et al. 2020; Valle et al. 2021b].
2. Analysis of Existing Reference Architectures: Publicly available reference architectures from various domains (e.g., Service-Oriented Architecture (SOA) [Arsanjani et al. 2007], microservices [Newman 2015], health information systems [Garcés 2018]) would be analyzed to extract interoperability-focused patterns and strategies [Garcés et al. 2021].
3. Expert Interviews and Surveys: Interviews with experienced software architects and system integrators would provide valuable practical insights into real-world interoperability challenges and the architectural solutions they employ [Valle et al. 2021]. Surveys could then be used to validate the identified solutions and gather input on their applicability and effectiveness.
4. Case Study Analysis: Examination of successful and unsuccessful interoperability projects would provide concrete examples of architectural solutions in action and highlight their strengths and weaknesses.

Solution Description and Categorization

Each identified architectural solution would be systematically documented using a standardized template. This template would include:

- Name and Aliases: A unique identifier and common alternative names for the solution.
- Problem Addressed: The specific interoperability challenge the solution aims to solve.
- Context: The conditions under which the solution is applicable.
- Solution Description: A detailed explanation of the architectural pattern or strategy, including its components, their interactions, and how they contribute to interoperability. Diagrams would be used where appropriate.
- Interoperability Layer(s) Addressed: Which specific layers (technical, syntactic, semantic, organizational) the solution primarily supports.
- Benefits: Advantages of applying the solution (e.g., reduced coupling, improved flexibility, enhanced data exchange).
- Drawbacks/Limitations: Disadvantages or challenges associated with the solution (e.g., increased complexity, performance overhead, vendor lock-in).
- Related Patterns/Solutions: How this solution relates to other architectural patterns or strategies.

- Examples of Use: Real-world or illustrative examples of the solution's application.
- References: Citations to relevant literature or case studies.

Solutions would be categorized based on their primary interoperability layer addressed (technical, syntactic, semantic, organizational) and further sub-categorized by common architectural paradigms (e.g., message-based, API-based, data-centric). This structured approach ensures navigability and comprehensiveness.

Validation and Refinement

The catalog would undergo an iterative validation and refinement process:

1. Peer Review: The initial compilation would be reviewed by a panel of software architecture and interoperability experts to ensure accuracy, completeness, and clarity.
2. Case Study Application: The catalog would be applied to analyze and design interoperability solutions in new hypothetical or real-world case studies to assess its practical utility and identify any gaps.
3. Feedback Incorporation: Feedback from reviews and application would be systematically incorporated to refine solution descriptions, categorization, and overall structure.

This methodological rigor aims to create a robust and valuable resource for advancing the practice of software system interoperability.

RESULTS

The conceptual result of this systematic approach is a structured Catalog of Interoperability Architectural Solutions (CIAS), organized to provide clear guidance for designing and implementing interoperable software systems. This section describes the anticipated structure and content of such a catalog, highlighting the types of solutions it would encompass across different interoperability layers.

Overall Structure of the Catalog

The CIAS would be organized primarily by the layers of interoperability, with each layer containing specific architectural patterns and strategies. This hierarchical organization would allow architects to quickly find solutions relevant to the particular interoperability challenge they face. A possible top-level structure could be:

- 1. Technical Interoperability Solutions
- 2. Syntactic Interoperability Solutions
- 3. Semantic Interoperability Solutions
- 4. Organizational/Pragmatic Interoperability Solutions
- 5. Cross-Layer and General Interoperability Strategies

Within each layer, solutions would be described using the template defined in the "Methods" section, ensuring consistency and comprehensive information for each entry.

Key Architectural Solutions by Interoperability Layer (Illustrative Examples)

1. Technical Interoperability Solutions

These solutions focus on enabling the fundamental exchange of data between heterogeneous systems at the network and protocol level.

- Remote Procedure Call (RPC) and RESTful APIs: Standard mechanisms for inter-process communication over networks. RESTful APIs, in particular, promote interoperability through their statelessness and use of standard HTTP methods, making them widely adopted for web-based integrations [Al-Zoubi and Wainer 2010].
- Message Brokers/Queues: Solutions like Enterprise Service Bus (ESB) or Message Queues (e.g., RabbitMQ, Apache Kafka) enable asynchronous, decoupled communication, abstracting away underlying network complexities. This allows systems to send and receive messages without direct knowledge of each other, fostering a more resilient integration [Repositorio 2021].
- Network Protocols Adapters: Components that translate between different network protocols to enable communication between systems using disparate communication standards.

2. Syntactic Interoperability Solutions

These solutions address the structural compatibility of data exchanged between systems, ensuring that data formats are understood and correctly parsed.

- Data Transformation Engines/Mediators: Components that convert data from one syntactic format to another (e.g., XML to JSON, CSV to proprietary binary format). These mediators are crucial in heterogeneous environments [Spalazzese and Inverardi 2010; Garcés et al. 2018b].
- Standardized Data Formats: Adopting industry-standard data formats (e.g., HL7 FHIR for healthcare [Benson and Grieve 2016], industry-specific XML schemas) ensures common understanding of data structure.
- Schema Registries: Centralized repositories for managing and sharing data schemas, ensuring that all systems adhere to agreed-upon data structures.

3. Semantic Interoperability Solutions

The most complex layer, focusing on ensuring that the meaning of exchanged information is unambiguously understood.

- Ontology-Based Mapping: Using formal ontologies to define shared concepts and relationships, and then mapping data elements from different systems to these common ontological terms [Aydin and Aydin 2020; Moreira et al. 2018]. This is critical for achieving true semantic understanding.
- Semantic Mediators/Brokers: Specialized mediators that not only transform data format but also reconcile semantic differences using semantic rules or reasoning engines.
- Controlled Vocabularies and Terminologies: Adoption of shared terminologies (e.g., SNOMED CT for healthcare [Benson and Grieve 2016]) to ensure consistent understanding of concepts.

4. Organizational/Pragmatic Interoperability Solutions

These solutions address the alignment of business processes, policies, and goals across different organizations or departments.

- Shared Business Process Models: Developing and agreeing upon common business process models that span multiple systems or organizations [Adamo et al. 2018].
- Choreography and Orchestration Patterns: Defining the sequence and coordination of interactions between multiple systems at a business process level [Benany and Beqqali 2018]. Choreography focuses on decentralized coordination, while orchestration involves a central coordinator.
- Joint Governance Frameworks: Establishing clear governance structures, policies, and agreements for data sharing and system integration across organizational boundaries.

5. Cross-Layer and General Interoperability Strategies

Beyond specific layer-focused solutions, the catalog would include overarching strategies applicable across multiple layers or concerning the broader system landscape.

- Reference Architectures: Domain-specific or general architectural blueprints that provide a common vocabulary and set of patterns for building interoperable systems (e.g., S3: Service-oriented Reference Architecture [Arsanjani et al. 2007]). These serve as guiding principles for architectural decision-making [Valle 2021].
- Adapter Pattern: A design pattern that allows the interface of an existing class to be used as another interface. This is a fundamental strategy for connecting incompatible systems at various levels [Maybee et al. 1996; Harrer et al. 2008].
- Decentralized Architectures (e.g., P2P, Blockchain): Architectures that inherently promote interoperability by distributing control and enabling direct communication between participants, reducing reliance on central authorities [Chainho et al. 2017; Chen 2018]. This is particularly relevant for achieving data source interoperability in ubiquitous enterprises [Pang et al. 2015].
- System-of-Systems Integration Approaches: Strategies specifically for managing the complexity of integrating independently managed, evolving systems into a larger cooperative system [Madni and Sievers 2014; Rezaei et al. 2014].
- Design Pattern Monitoring: Techniques for ensuring that implemented systems adhere to the intended architectural patterns for interoperability [Hallstrom et al. 2006].

The catalog would present these solutions with detailed descriptions, pros and cons, and examples, allowing practitioners to select the most appropriate strategy for their specific interoperability challenges. The illustrative quantitative results, if available from empirical studies based on the catalog's application, would show the effectiveness of applying specific architectural solutions in improving interoperability metrics (e.g., reduction in integration effort, improved data consistency).

DISCUSSION

The conceptualization and anticipated structure of the Catalog of Interoperability Architectural Solutions (CIAS) highlight its potential to significantly advance the practice of building interoperable software systems. By systematically organizing architectural patterns and strategies across various interoperability layers, the CIAS addresses a critical need in an increasingly interconnected software landscape.

The primary benefit of such a catalog lies in its ability to demystify interoperability. Often perceived as a vague and complex challenge, interoperability can be broken down into concrete architectural problems, each with a set of proven solutions. This structured approach empowers software architects and engineers by providing them with a common language and a toolkit to identify, design, and implement effective integration strategies. For instance, understanding the nuances between technical, syntactic, and semantic interoperability [Maciel et al. 2024] allows for more targeted architectural interventions.

The CIAS would foster architectural decision-making excellence [Valle 2021]. Instead of ad-hoc solutions, architects could leverage documented patterns, understanding their context, benefits, drawbacks, and real-world applicability. This reduces the risk of costly rework and improves the overall quality of integration efforts. The catalog's detailed descriptions, including problem statements and potential side effects, enable informed choices, promoting best practices and reusability of successful integration approaches. For example, knowing

when to choose a message broker over direct API calls, or when to invest in semantic mapping, becomes clearer with a comprehensive guide.

Furthermore, a standardized catalog contributes to knowledge transfer and education. It provides a structured curriculum for teaching and learning about interoperability in software architecture. New team members can quickly grasp established patterns, and experienced practitioners can discover novel approaches or reinforce their understanding of existing ones. This is particularly valuable given the persistent challenges in achieving interoperability in diverse domains [Abukwaik and Rombach 2017; Daliya and Ramesh 2019; Farooq et al. 2020].

Challenges and Limitations

Despite its significant potential, the development and maintenance of such a catalog present several challenges:

1. Completeness and Evolution: The software landscape is constantly evolving, with new technologies, protocols, and architectural styles emerging regularly. Maintaining a comprehensive and up-to-date catalog requires continuous effort to identify and document new solutions. What constitutes "interoperability" itself can also evolve with advancements like IoT [Noura et al. 2019; Rahman and Hussain 2020] or decentralized systems [Chen 2018].
2. Context-Dependency: Architectural solutions are rarely universally applicable; their effectiveness often depends heavily on the specific context, including system size, performance requirements, security needs, and organizational culture. While the catalog aims to document context, applying solutions still requires significant architectural expertise.
3. Validation and Effectiveness: Quantitatively demonstrating the effectiveness of architectural solutions in improving interoperability can be challenging. Empirical studies are needed to validate the proposed solutions and measure their impact in real-world scenarios.
4. Granularity: Determining the appropriate level of granularity for documenting solutions is crucial. Too broad, and they lack practical guidance; too fine-grained, and the catalog becomes unwieldy.
5. Adoption: The ultimate value of the catalog depends on its widespread adoption by practitioners. This requires intuitive organization, easy accessibility, and active promotion within the software engineering community.

Future Directions

Future work related to the CIAS could include:

1. Tool Support: Developing tools that integrate with the catalog, perhaps offering recommendations for interoperability solutions based on system requirements or automating the generation of architectural diagrams for selected patterns.
2. Empirical Validation: Conducting extensive empirical studies and case studies to validate the effectiveness of the catalog's solutions in improving interoperability metrics across different domains and system types.
3. Community Contribution Model: Establishing a community-driven model for maintaining and expanding the catalog, allowing practitioners to contribute new patterns, refine existing ones, and share their experiences. This could mirror successful open-source pattern repositories.
4. Integration with Reference Architectures: Tightly integrating the catalog with existing and emerging reference architectures [Garcés et al. 2021], providing concrete architectural choices within those broader frameworks.
5. Quantitative Metrics: Defining and collecting more quantitative metrics related to the application of interoperability solutions, such as reduction in integration time, error rates, or maintenance costs. This would further support evidence-based architectural decision-making.

In conclusion, a well-structured and comprehensive Catalog of Interoperability Architectural Solutions offers a promising path towards systematically addressing the complex challenge of software system interoperability. By providing a curated repository of architectural knowledge, it can empower practitioners, streamline development processes, and ultimately contribute to the creation of more robust, flexible, and integrated software ecosystems.

REFERENCES

[1] Abukwaik, H., Rombach, D.: “Software interoperability analysis in practice: A survey”; International Conference on Evaluation and Assessment in Software Engineering (EASE), ACM (2017), 12–20.

[2] Adamo, G., Borgo, S., Di Francescomarino, C., Ghidini, C., Guarino, N.: “On the notion of goal in business process models”; International Conference of the Italian Association for Artificial Intelligence, Springer (2018), 139–151.

[3] Al-Zoubi, K., Wainer, G.: “Rise: Rest-ing heterogeneous simulations interoperability”; Proceedings of the 2010 Winter Simulation Conference (2010), 2968-2980.

[4] Arsanjani, A., Zhang, L.-J., Ellis, M., Allam, A., Channabasavaiah, K.: “S3: A service-oriented reference architecture”; IT professional, 9 (2007), 10–17.

[5] Aydin, S., Aydin, M. N.: “Semantic and syntactic interoperability for agricultural open-data platforms in the context of IoT using crop-specific trait ontologies”; Applied Sciences, 10, 13 (2020), 4460.

[6] Baldwin, W. C., Sauser, B. J., Boardman, J.: "Revisiting "The Meaning of Of" as a Theory for Collaborative System of Systems"; IEEE Systems Journal, 11, 4 (2017), 2215-2226.
[7] Bass, L.: "Software architecture in practice"; Addison-Wesley, Massachusetts, USA (2013).
[8] Benany, E., Beqqali, E.: "Choreography for interoperability in the e-Government applications"; International Conference on Intelligent Systems and Computer Vision (ISCV), IEEE (2018), 1–4.
[9] Benson, T., Grieve, G.: "Principles of health interoperability: SNOMED CT, HL7 and FHIR"; Springer, London, UK (2016).
[10] Bicer, V., Laleci, G. B., Dogac, A., Kabak, Y.: "Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain"; ACM, New York, USA, 34, 3 (2005).
[11] Bouziat, T., Camps, V., Combettes, S.: "A Cooperative SoS Architecting Approach Based on Adaptive Multi-agent Systems"; International Workshop on Software Engineering for Systems-of-Systems (SESoS), ACM (2018), 8–16.
[12] Burns, T., Cosgrove, J., Doyle, F.: "A Review of Interoperability Standards for Industry 4.0."; Procedia Manufacturing, 38 (2019), 646–653.
[13] Chainho, P., Drüsedow, S., Pereira, R. L., Chaves, R., Santos, N., Haensge, K., Portabales, A. R.: "Decentralized Communications: Trustworthy interoperability in peer-to-peer networks"; 2017 European Conference on Networks and Communications (EuCNC) (2017), 1-5.
[14] Chen, J.: "Devify: Decentralized Internet of Things Software Framework for a Peer-to-Peer and Interoperable IoT Device"; ACM, New York, USA, 15, 2 (2018).
[15] Chen, D., Doumeingts, G., Vernadat, F.: "Architectures for enterprise integration and interoperability: Past, present and future"; Computers in Industry, 59, 7 (2008), 647–659.
[16] Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., Little, R.: "Documenting software architectures: views and beyond"; Pearson Education (2002).
[17] Daliya, V. K., Ramesh, T. K.: "Data Interoperability Enhancement of Electronic Health Record data using a hybrid model"; International Conference on Smart Systems and Inventive Technology (2019), 318-322.
[18] Diván, M., Sánchez Reynoso, M. L.: "Fostering the Interoperability of the Measurement and Evaluation Project Definitions in PAbMM"; International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (2018), 231-238.
[19] eHGI: "Discussion paper on semantic and technical interoperability" (2017).
[20] Farooq, M. O., Wheelock, I., Pesch, D.: "IoT-Connect: An Interoperability Framework for Smart Home Communication Protocols"; IEEE Consumer Electronics Magazine, 9, 1 (2020), 22-29.
[21] Garcés, L.: "A Reference Architecture for Healthcare Supportive Home (HSH) systems"; Universidade de São Paulo (2018).
[22] Garcés, L., Oquendo, F., Nakagawa, E.: "Towards a Taxonomy of Software Mediators for Systems-of-Systems"; Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS) (2018), 53–62.
[23] Garcés, L., Martínez-Fernández, S., Oliveira, L., Valle, P., Ayala, C., Franch, X., Nakagawa, E. Y.: "Three decades of software reference architectures: A systematic mapping study"; Journal of Systems and Software, 179 (2021), 111004.
[24] Garlan, D.: "Software architecture"; Wiley Encyclopedia of Computer Science and Engineering (2007).
[25] Gazzarata, R., Giannini, B., Giacomini, M.: "A SOA-based platform to support clinical data sharing"; Journal of healthcare engineering, 2017 (2017).
[26] Guo, Y., Hu, Y., Afzal, J., Bai, G.: "Using P2P technology to achieve eHealth interoperability"; International Conference on Service Systems and Service Management (2011), 1-5.
[27] Hallstrom, J. O., Dalton, A. R., Soundarajan, N.: "Parallel Monitoring of Design Pattern Contracts."; SEKE, Citeseer (2006), 236–241.
[28] Harrer, A., Pinkwart, N., McLaren, B. M., Scheuer, O.: "The Scalable Adapter Design Pattern: Enabling Interoperability Between Educational Software Tools"; IEEE Transactions on Learning Technologies, 1, 2 (2008), 131-143.
[29] Ibrahim, N., bin Hassan, M.: "A survey on different interoperability frameworks of SOA systems towards seamless interoperability"; International Symposium in Information Technology (ITSim), IEEE (2010), 1119–1123.
[30] IEEE: "The Authoritative Dictionary of IEEE Standards Terms"; IEEE Std 100, 2000 (2000), 1–1362.
[31] Ingram, C., Payne, R., Fitzgerald, J.: "Architectural Modelling Patterns for Systems of Systems"; Annual International Council on Systems Engineering (INCOSE), Wiley Online Library (2015), 1177–1192.
[32] Keshav, R., Gamble, R.: "Towards a taxonomy of architecture integration strategies"; International Workshop on Software Architecture (ISAW), ACM (1998), 89–92.
[33] Kubicek, H., Cimander, R., Scholl, H. J.: "Chapter 7 - Layers of interoperability"; Organizational Interoperability in E-Government (ICSOC), Springer (2011), 85–96.
[34] Maciel, R. S. P., David, J. M. N., Claro, D., Braga, R.: "Full interoperability: Challenges and opportunities for future information systems"; Sociedade Brasileira de Computação (2017).
[35] Maciel, R., Valle, P. H. D., Santos, K., Nakagawa, E. Y.: "Systems Interoperability Types: A Tertiary Study"; ACM Computing Survey, 56, 10 (2024), 1–37.
[36] Madni, A. M., Sievers, M.: "System of systems integration: Key considerations and challenges"; Systems Engineering, 17, 3 (2014), 330–347.

[37] Maybee, M. J., Heimbigner, D. M., Osterweil, L. J.: "Multilanguage interoperability in distributed systems. Experience report"; International Conference on Software Engineering (1996), 451-463.
[38] Moreira, M. W. L., Rodrigues, J. J. P. C., Sangaiah, A. K., Al-Muhtadi, J., Korotaev, V.: "Semantic interoperability and pattern classification for a service-oriented architecture in pregnancy care"; Future Generation Computer Systems 89 (2018), 137–147.
[39] Muketha, G. M., Wamocho, L., Micheni, E.: "A Review of Agent Based Interoperability Frameworks and Interoperability Assessment Models"; Scholars Journal of Engineering and Technology (SJET), 2 (2014).
[40] Newman, S.: Building microservices: designing fine-grained systems";O'Reilly Media, Inc.", New York, USA (2015).
[41] Noura, M., Atiquzzaman, M., Gaedke, M.: "Interoperability in internet of things: Taxonomies and open challenges"; Mobile Networks and Applications, 24, 3 (2019), 796–809.
[42] Pang, L. Y., Zhong, R. Y., Fang, J., Huang, G. Q.: "Data-source interoperability service for heterogeneous information integration in ubiquitous enterprises"; Advanced Engineering Informatics, 29, 3 (2015), 549–561.
[43] Rahman, H., Hussain, M. I.: "A comprehensive survey on semantic interoperability for Internet of Things: State-of-the-art and research challenges"; Transactions on Emerging Telecommunications Technologies, 31, 12 (2020).
[44] Repositorio: "Repositório Online de Padrões: Enterprise Integration Patterns"; http://www.enterpriseintegrationpatterns.com (2021).
[45] Rezaei, R., Chiew, T., Lee, S. P.: "An interoperability model for ultra large scale systems"; Advances in Engineering Software, 67 (2014), 22–46.
[46] Spalazzese, R., Inverardi, P.: "Mediating connector patterns for components interoperability"; 4th European Conference on Software Architecture (ECSA), Springer (2010), 335–343.
[47] Valle, P. H. D.: "Architectural decision-making on interoperability in software-intensive systems"; Universidade de São Paulo (2021).
[48] Valle, P., Garcés, L., Nakagawa, E.: "A Typology of Architectural Strategies for Interoperability"; 13th Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS) (2019), 3-12.
[49] Valle, P. H. D., Garcéss, L., Guessi, M., Martínez-Fernández, S., Nakagawa, E. Y.: "Approaches for Describing Reference Architectures: A Systematic Mapping Study"; XXIII Iberoamerican Conference on Software Engineering (CIbSE), Springer (2020), 1–14.
[50] Valle, P. H. D., Garcés, L., Nakagawa, E. Y.: "Architectural Strategies for Interoperability of Software-Intensive Systems: Practitioners' Perspective"; ACM Symposium on Applied Computing, Track Software Architecture: Theory, Technology, and Applications (SAC/SATTA 2021), ACM (2021), 1–10.
[51] Valle, P. H. D., Garcés, L., Volpato, T., Martínez-Fernández, S., Nakagawa, E. Y.: "Towards Suitable Description of Reference Architectures"; PeerJ Computer Science (2021), 1-26.
[52] Valle, P. H. D., Tonon, V. R., Garcés, L., Rezende, S. O., Nakagawa, E. Y.: "TASIS: A typology of architectural strategies for interoperability in software-intensive systems"; Computer Standards amp; Interfaces, 91 (2025), 103874.
[53] van der Veer, H., Wiles, A.: "Achieving technical interoperability"; European telecommunications standards institute, 1 (2008).