Architectural Practices for Future Trends in Mission-Critical Systems

Gustavo S. Melo[1], **Michel S. Soares**[1]

[1]Federal University of Sergipe – São Cristóvão – SE – Brazil

gustavo.dsantosmelo@gmail.com, michel@dcomp.ufs.br

Abstract. *Critical software systems require high fault tolerance and the ability to adapt to technological and regulatory changes. The increasing complexity of these systems makes it difficult to balance quality attributes such as performance, security, and maintainability. Software architecture is essential to ensure quality attributes and continuous evolution. This article reviews software architecture practices that strengthen quality attributes in critical systems. Results show advances in resilience and automation, highlighting the adoption of DevOps, AI-based predictive maintenance, and quantum computing security, composing a roadmap for modern mission-critical Systems.*

1. Introduction

The evolutionary path of the software industry has reached new levels of knowledge, making it possible to create solutions with more rigorous processes, significantly increasing the probability of a software's success [Agerwala and Bass 2024]. One of the determining factors for this success is intrinsically linked to software architecture, which is considered an essential element for the sustainability and evolution of software. This is because, when poorly designed, software may not meet future demands or keep up with the dynamic needs of industry, compromising its longevity and relevance [Biffl et al. 2023].

Software architecture assessment is therefore one of the pillars of the development lifecycle, enabling the measurement of the level of knowledge and planning among the involved teams [Kaur et al. 2021]. For mission-critical systems, the architecture must consider not only functional and quality requirements, but also the essential objectives and values that underpin the operation of the mission. In such cases, the effectiveness of the critical elements is assessed based on the level of maturity achieved and the knowledge available, factors that are important for the accurate prediction of results and the continued success of the mission.

This article presents an analysis of software architecture in mission-critical systems, starting with fundamental concepts and their importance, moving on to specific characteristics of these systems, architectural aspects, evaluation methods, requirements elicitation processes and risk management strategies, culminating in discussions of future trends. This structure allows for a progressive understanding of the subject, from its foundations to its practical applications and emerging challenges, offering both a theoretical and practical view of the development and maintenance of critical systems.

2. Software Architecture for Mission-Critical Systems

Software architecture is defined as an abstraction that describes the components of a software system and their interactions, allowing a structural view that facilitates the under-

standing, analysis and evolution of software. Architecture not only involves the organisation of components, but also reflects the design decisions that affect quality and maintainability of software. The success of software is based on the implementation of its architecture, the analysis of trade-offs, a deep understanding of the product, the elicitation of its requirements and the skills of developers [Di Pompeo and Tucci 2023]. A well-defined software architecture guides design decisions, facilitates communication between teams, mitigates technical risks and promotes essential qualities such as performance, scalability, security, maintainability and testability.

New technological approaches, such as DevOps and microservice-based architectures, have increased the complexity and challenges of evaluating architectures in mission-critical systems. According to Zhang [Li et al. 2021], the integration of continuous development and operation practices (DevOps) in critical systems requires architectural evaluations that include not only runtime performance but also maintainability and resilience in rapidly changing environments.

Table 1 presents a classification of common risks in mission-critical systems, along with mitigation strategies and recommended technologies to minimise the potential impact of each risk. The categorisation covers key aspects such as system security, technical failures, operational risks, resilience and budget constraints. Each type of risk has a description of the impact it can have and suggested approaches to mitigate it, such as redundancy, backups and regular audits.

Table 1. Risk Classification and Mitigation Strategies in Mission Critical Systems

Type of Risk	**Potential Impact**	**Mitigation Strategy**	**Example Tools/Technologies**
Systems Security	Compromise of data and operations	Implementation of multi-factor authentication, encryption, and firewalls	Firewalls, intrusion detection systems
Technical failure	Interruption of critical functionalities	Continuous monitoring, preventive maintenance and contingency plans	System monitoring software, failover plans
Operational Risk	Dependence on external components and vulnerability to third-party failures	Regular audits, redundancy and supplier audits	Backup systems, network redundancy
Recovery Capacity	Prolonged downtime due to system failure	Implementation of backups, disaster recovery plans and data replication	Cloud storage, incremental backup
Budget and Economic Constraints	Limitation on developing and maintaining security strategies	Strategic resource allocation and budget planning for regular upgrades	Budget management systems, financial planning

In this way, one can highlight the main points for building a robust software architecture for mission-critical systems, such as:

1. **Requirements Definition and Analysis**
    - Requirements Elicitation
    - Critical Requirements Identification
    - Detailed Requirements Documentation
2. **Architectural Design**
    - Selection of Architectural Styles
    - Definition of Critical Modules and Components
    - Identification of Fault Tolerance and Redundancy Mechanisms
3. **Architectural Implementation**
    - Development of Components Following Architectural Requirements
    - Integration of Subsystems
    - Ensuring Adherence to Security Standards and Practices
4. **Architectural Testing and Requirements Validation**
    - Performance and stress testing to assess robustness
    - Verification and validation of critical requirements
    - Failure and recovery testing (failover, redundancy)
5. **Risk Management and Failure Mitigation**
    - Continuous risk analysis
    - Implementation of mitigation strategies (such as backups, redundancies)
    - Security updates and vulnerability analysis
6. **Continuous Assessment and Maintenance**
    - Monitoring of the production system
    - Adjustments and improvements based on feedback
    - Preventative maintenance and updates for compatibility and security

3. Future Trends

The evolution of mission-critical systems is closely linked to the emergence of new technological paradigms and the growing demand for security, resilience, and scalability [Liao 2024]. These systems, often responsible for supporting essential services such as healthcare, finance, transportation, and energy, cannot afford downtime or failures, as their unavailability can result in serious social and economic impacts. The following sections discuss technological trends that are reshaping software architectures.

3.1. Artificial Intelligence in Software Architecture Assessment

Artificial intelligence (AI) has shown significant potential in helping with the evaluation of software architecture, transforming traditional approaches, and offering new possibilities for optimisation, consistency, and decision-making. AI tools, such as those based on Large Language Models (LLMs), are increasingly integrated into software development workflows, redefining the way architects approach modern design [Ramachandran 2025].

Traditionally, software architects employ a variety of tools to create architectural designs, including UML diagrams. Assisted AI tools such as GitHub Copilot, ChatGPT and BlackBoxAI are being evaluated for their ability to generate these designs based on contextual prompts [Ramachandran 2025]. Applying AI to software architecture evaluation can improve productivity, ensure design consistency, and support architectural decision-making [Ramachandran 2025]. Studies have explored the effectiveness of these tools in generating diagrams, with comparative analyses seeking to determine

how closely the results generated by AI align with established architectural principles [Ramachandran 2025].

AI-based tools will automate routine tasks, optimise workflows and resource allocation, allowing software engineers to focus on creative problem solving. Future research should also address the generation of large, high-quality and unbiased datasets for training AI models, as well as the development of reliable benchmarks for an unbiased evaluation of all Software Engineering tasks [Ahmed et al. 2025]. Collaboration between humans and AI in Software Engineering teams will require an understanding of how to interpret AI-based decisions and recommendations [Ahmed et al. 2025].

3.2. 5G and High-Speed Networks

The advancement of 5G networks, and their evolution to Beyond 5G, is one of the central pillars for defining new trends in software architecture. This technology is not limited to increasing transmission rates, but also enables an ecosystem of distributed applications that require low latency, high reliability, and support for extreme mobility. For software architecture, this means designing systems that are resilient, adaptive, and capable of integrating with dynamic network infrastructures.

Studies on communications in high-mobility scenarios [Han et al. 2019] demonstrate how beamforming techniques, channel prediction, and the use of location data can significantly increase spectral efficiency. Although focused on the context of high-speed trains, these principles have broader implications: distributed system architectures could benefit from similar strategies to ensure quality of service in dense urban environments, autonomous vehicles, or large-scale sensor networks. Similarly, the work of Noh et. al, [Noh et al. 2020] demonstrates that 5G NR incorporates elements designed to handle scenarios of intense variability, such as flexible OFDM (Orthogonal Frequency Division Multiplexing) numerologies, adaptive reference signals, and new handover mechanisms with near-zero interruptions. This technical capability reinforces 5G's role as an infrastructure on which software architectures can implement critical services, from real-time telemedicine to IoT-based industrial applications, requiring new standards of interoperability, elasticity, and continuous monitoring.

A relevant aspect for software architects is the convergence between 5G, edge computing, and distributed cloud computing. With the decentralization of edge processing, it becomes necessary to design architectures capable of orchestrating components across multiple network layers, balancing loads, tolerating failures, and ensuring end-to-end security. Furthermore, in mission-critical systems such as industrial automation, digital healthcare, or smart transportation, sub-millisecond latency and massive device communication open up space for event-driven and self-adaptive architectures that adjust according to the state of the network. In this sense, standards such as ISO/IEC/IEEE 42020:2019 can play a central role in methodological adaptation to incorporate the emerging requirements of these new infrastructures [Junior et al. 2019].

3.3. Cloud Computing

The evolution of cloud computing has been reshaping the way software systems are designed, deployed, and maintained. One of the key points is the integration between cloud computing and intelligent industrial systems, especially in Industry 4.0

[Steinmetz et al. 2018]. The adoption of cloud platforms in production lines, combined with advanced algorithms such as fuzzy PID, enables not only remote and real-time monitoring but also intelligent and predictive decision-making [Liu 2024]. This trend reinforces that, in the future, the cloud will be the backbone for autonomous and highly adaptive industrial architectures.

In the fields of software engineering and legacy systems reengineering, the cloud represents a strategic opportunity for application modernization [Raja et al. 2023]. This makes it possible to gradually migrate legacy systems or completely restructure them on service-based platforms, ensuring greater scalability, interoperability, and quality of service (QoS). A key trend is the consolidation of tools and methodologies that facilitate this migration process, reducing the costs and risks associated with replacing critical systems still in operation. Besides technical advances, cloud computing also stands out for its economic benefits compared to maintaining on-premises servers. The cloud-based model allows organizations to reduce infrastructure costs by avoiding high initial investments in hardware, energy, and maintenance [Raja et al. 2023].

More broadly, future trends in cloud computing point to three major movements: (1) sector specialization, with architectures adapted to industries such as healthcare, automotive, and manufacturing, (2) integration with edge computing, expanding the distributed hybrid computing paradigm, and (3) strengthening cloud-oriented software engineering, including reengineering, process automation, and greater standardization in critical architectures. Such advances not only reinforce the cloud's role as a global infrastructure but also expand its relevance in supporting critical systems.

3.4. Edge Computing and IoT Integration

Edge computing emerges as a direct response to the limitations of the centralized cloud computing model, especially in Internet of Things (IoT) scenarios and latency-sensitive applications. As billions of connected devices begin to generate data on a large scale, the need for processing close to the source becomes essential. This approach not only reduces bandwidth overhead on central servers but also enables near-real-time responses, an indispensable requirement for critical and distributed systems [Dayoub 2025].

In the IoT domain, integration with edge computing architectures has proven strategic. In contrast to the traditional three-tier model (sensing, network, and cloud), edge-based architectures add an additional layer responsible for local data preprocessing and analysis. This results in greater energy efficiency, lower latency, and increased system resilience. Recent studies demonstrate that, in mobility scenarios, the use of platforms such as AdvantEDGE reduces latency to approximately 30–50 ms, demonstrating the effectiveness of Multi-access Edge Computing (MEC) in time-sensitive IoT applications [Dayoub 2025].

Another emerging aspect is the integration of Edge Computing and blockchain, aiming to address the challenges of security, privacy, and resource allocation in distributed IoT systems. The use of blockchain in conjunction with Mobile Edge Computing (MEC) allows for the creation of decentralized consensus mechanisms to manage authentication, storage, and data sharing [Bai et al. 2022]. Furthermore, intelligent resource allocation algorithms can optimize both energy consumption and latency, contributing to more scalable and reliable architectures in large-scale environments.

These discussions reinforce the need for software architecture to evolve toward hybrid and decentralized models, in which cloud and edge work seamlessly. This imposes new demands on software architects: dealing with device heterogeneity, designing context-aware adaptive systems, and ensuring interoperability between multiple processing layers. The result is a transformation of the architectural paradigm, in which the edge is not just a supporting resource, but a strategic element for sustaining the next generation of distributed and mission-critical systems.

3.5. Security and Cyber-Resilience

Future software architectures must evolve beyond merely delivering functionality, integrating security and resilience as fundamental pillars from their inception. This paradigm shift requires a clear distinction between cybersecurity and cyberresilience. While cybersecurity focuses on protecting IT assets, such as data, cyberresilience is defined as the ability of a system or organization to defend itself against successful cyberattacks and revert to a normal operational state when defenses fail [Alhidaifi et al. 2024]. Therefore, the architecture of the future can no longer treat security as an external layer or an afterthought where it must be designed with the assumption that failures will occur, focusing on the system's ability to prepare for, absorb, recover from, and adapt to adverse events that affect business operations [Alhidaifi et al. 2024].

To manage the complexity inherent in Systems of Systems, future architectures will require strategies that reduce the scope of the problem. By focusing on operational intent and acceptable behaviors within well-defined boundaries, developers can implement an advanced form of defensive programming. This allows for more effective design and testing of resilience, ensuring that, even in the face of unforeseen parameters or scenarios, the system behaves as expected, allowing it to degrade gradually and safely rather than fail catastrophically [Mison et al. 2024].

However, designing for resilience is not enough if its effectiveness cannot be proven. One of the most critical and urgent trends is the development and adoption of objective and quantifiable security metrics. Currently, the software industry lacks a reliable system for measuring security, leading to procurement and architecture decisions based on vendor claims, anecdotal reports, or personal experience rather than hard data [Eggendorfer and Andresen 2024]. Software architecture needs standardized methods for assessing and comparing the security level of different components and systems. The existence of such metrics would not only empower architects to make informed decisions but also create a market incentive for vendors to genuinely invest in more secure products, increasing collective cyber resilience.

3.6. Microservices Architectures

A dominant trend in modern software architecture is the decomposition of legacy monolithic applications into microservices architectures, driven by the promise of greater scalability, agility, and failure resilience [Souza et al. 2023]. However, this transition is a complex and expensive undertaking, still in its early stages of maturity. Although several approaches have been proposed, consolidated methods for combining different types of analysis data, such as static, dynamic, and evolutionary data, are lacking, and automated tool support remains insufficient [Abgaz et al. 2023].

The complexity introduced by the distributed and heterogeneous nature of microservices poses significant challenges not only in decomposition but also in software testing. This indicates a lack of empirical validation and industry-grade benchmarks that could facilitate wider adoption of robust testing techniques. Research is dominated by laboratory experiments, and there are notable gaps in areas such as flexibility testing and, surprisingly, security testing, which were not directly addressed by any of the primary studies analyzed [Abgaz et al. 2023]. Even after decomposition, ensuring a system's ability to be adapted or extended efficiently remains a critical challenge [Bogner et al. 2021]. Practices such as defining architectural principles, test automation, and the use of patterns are considered crucial. However, most industry tools and metrics still focus on the code quality of individual services, neglecting macro-architectural assessment.

The most obvious gap in industrial practice is the lack of tools and metrics for architectural assessment, even when the most important challenges, such as defining service boundaries and integration, are architectural in nature. This is reflected in testing challenges, where system testing emerges as the most investigated level, followed by integration testing, indicating the significant focus on validating the behavior of the entire system and the interaction between services. The predominance of specification-based strategies in functional testing and the lack of focus on quality attributes such as security and flexibility reinforce the need for research and development of more comprehensive, macro-architectural validation approaches [Bogner et al. 2021].

3.7. Quantum Computing and Post-Quantum Security

The rise of quantum computing represents a paradigm shift that fundamentally redefines Software Engineering, requiring a complete rethinking of its practices, especially in the area of Requirements Engineering (RE). Unlike classical software development, quantum software is intrinsically linked to quantum mechanics, introducing a new level of abstraction and a reliance on properties such as superposition and entanglement. This unique characteristic, where operations are inherently probabilistic, renders traditional RE methods inadequate. The threat becomes even more concrete with "harvest now, decrypt later" attacks, where adversaries capture data encrypted today to decrypt it in the future with quantum computers, posing an immediate risk to long-term sensitive data such as health records [Sepúlveda et al. 2024, Ezeogu 2025].

One of the central challenges in RE for quantum computing (QRE) is the difficulty in defining specific requirements for systems that must be resilient to quantum attacks. Most practical applications will be hybrid systems, integrating quantum and classical components. This requires specifying complex interactions between different computational paradigms and choosing between various quantum security technologies, such as lattice-based cryptography (LBC), quantum key distribution (QKD), and quantum hash functions (QHF). The lack of established standards for RE in this domain exacerbates the problem, hindering consistent requirements collection, analysis, and validation [Sepúlveda et al. 2024].

Another significant challenge is the knowledge gap and rapid technological evolution. There is a shortage of requirements engineers trained in the principles of quantum computing, which limits the ability to effectively translate quantum capabilities into software requirements. This need for readiness is reinforced by the urgency to develop

"quantum literacy" among IT professionals so they can make informed decisions about adopting new technologies and security strategies [Sepúlveda et al. 2024, Ezeogu 2025]. Integrating security from the early stages is a paramount concern. Quantum computing threatens classical cryptographic algorithms such as RSA and ECC, which are vulnerable to attacks using Shor's algorithm. This requires a transition to post-quantum cryptography (PQC), with the adoption of NIST-standardized algorithms such as CRYSTALS-Kyber, CRYSTALS-Dilithium, FALCON, and SPHINCS+. A key architectural trend for managing this migration is "crypto-agility", which enables a smooth transition between traditional and post-quantum cryptographic methods, mitigating risks and ensuring operational continuity.

In the future, formalizing QRE processes, empirically validating PQC algorithms, and standardizing architectural practices will be crucial. Implementing PQC introduces performance overheads in latency, storage, and compute, which should be considered as critical non-functional requirements during design. For example, studies show that Kyber-1024 is suitable for cryptographic tasks with minimal overhead, while Dilithium-5 offers a good balance for long-term signature security. Furthermore, the integration of quantum security into CI/CD pipelines, known as Quantum Secure DevOps (QSecOps), faces barriers such as a lack of standardization and the complexity of key management, indicating that the path forward will require both technical innovation and the development of new professional skills [Ezeogu 2025].

4. Conclusion

Software development has grown exponentially compared to its early days. Practices, decision-making, trade-off analysis, new technologies that are constantly emerging, the globalization of development teams, and many other factors have contributed to positively shaping the current software development landscape. Mission-critical systems incorporate all of these factors, but with an additional complexity: the requirement that these systems cannot fail. The failure of a system of this magnitude can trigger a financial collapse or, in more delicate contexts, lead to the loss of lives. As such, these systems are subjected to the most rigorous evaluation, validation, and development processes in the software industry to ensure reliability, safety, and robustness at an exceptional level.

Emerging trends and challenges are also highly relevant due to the benefits they promise. This technological advancement is increasingly consolidating itself as a strategic ally for mission-critical systems, not only for predicting and mitigating potential failures, but also for optimizing processes, analyzing large volumes of data in real time, and even facilitating autonomous decision-making in critical situations.

Technologies such as edge computing and advanced cybersecurity have played a central role in the evolution of these systems. Edge computing enables data processing closer to the source, reducing latency and increasing efficiency for systems that require immediate responses. Advanced cybersecurity, in turn, is becoming increasingly essential in an environment where cyberattacks are becoming more sophisticated and frequent. New architectural models, such as distributed and microservices-based architectures, also contribute to the resilience and scalability of systems, although they bring challenges in integration and complexity management.

Shortly, the development of mission-critical systems will require a careful balance

between incorporating new technologies and maintaining strict security and reliability standards. A commitment to innovation, combined with caution and technical rigour, will continue to be the basis for these systems to evolve without compromising their essential qualities.

References

Abgaz, Y., McCarren, A., Elger, P., Solan, D., Lapuz, N., Bivol, M., Jackson, G., Yilmaz, M., Buckley, J., and Clarke, P. (2023). Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review. *IEEE Transactions on Software Engineering*, 49(8):4213–4242.

Agerwala, G. and Bass, L. (2024). Teaching Software Architecture Design - Building Intuition. In *Proc. of the 1st Int. Workshop on Designing Software*, pages 27—-33, New York, NY, USA. ACM.

Ahmed, I., Aleti, A., Cai, H., Chatzigeorgiou, A., He, P., Hu, X., Pezzè, M., Poshyvanyk, D., and Xia, X. (2025). Artificial Intelligence for Software Engineering: The Journey So Far and the Road Ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5).

Alhidaifi, S. M., Asghar, M. R., and Ansari, I. S. (2024). A Survey on Cyber Resilience: Key Strategies, Research Challenges, and Future Directions. *ACM Comput. Surv.*

Bai, Z., Wan, J., Li, L., Liu, C., and Duan, M. (2022). Resource Allocation of IoT systems Integrated with Blockchain and Mobile Edge Computing. In *2022 4th Int. Conf. on Frontiers Technology of Information and Computer*, pages 377–382.

Biffl, S., Navarro, E., Mirandola, R., and Weyns, D. (2023). Architecting for a Sustainable Digital Society. *J. of Systems and Software*, 200:111668.

Bogner, J., Fritzsch, J., Wagner, S., et al. (2021). Industry Practices and Challenges for the Evolvability Assurance of Microservices. *Empirical Software Engineering*, 26(104).

Dayoub, A. J. (2025). The Integration of Edge Computing into IoT Application Using AdvantEDGE Platform, Case Study: Mobility. In *2025 7th Int. Youth Conf. on Radio Electronics, Electrical and Power Engineering*, pages 1–5.

Di Pompeo, D. and Tucci, M. (2023). Quality Attributes Optimization of Software Architecture: Research Challenges and Directions. In *2023 IEEE 20th Int. Conf. on Software Architecture Companion*, pages 252–255.

Eggendorfer, T. and Andresen, K. (2024). Using Security Metrics to improve Cyber-Resilience. In *Proceedings of the IARIA Congress*, pages 152–157.

Ezeogu, A. O. (2025). Post-Quantum Cryptography for Healthcare: Future-Proofing Population Health Databases Againts Quantum Computing Threats. *Research Corridor Journal of Engineering Science*, 2(1):29–56.

Han, S., I., C.-L., Xie, T., Wang, S., Huang, Y., Dai, L., Sun, Q., and Cui, C. (2019). Achieving High Spectrum Efficiency on High Speed Train for 5G New Radio and Beyond. *IEEE Wireless Communications*, 26(5):62–69.

Junior, A. A. C., Misra, S., and Soares, M. S. (2019). ArchCaMO - A Maturity Model for Software Architecture Description Based on ISO/IEC/IEEE 42010:2011. In Misra,

S., Gervasi, O., Murgante, B., Stankova, E., Korkhov, V., Torre, C., Rocha, A. M. A., Taniar, D., Apduhan, B. O., and Tarantino, E., editors, *Computational Science and Its Applications – ICCSA 2019*, pages 31–42. Springer International Publishing.

Kaur, K., Khurana, M., and Manisha (2021). Impact of Agile Scrum Methodology on Time to Market and Code Quality – A Case Study. In *2021 3rd Int. Conf. on Advances in Computing, Communication Control and Networking*, pages 1673–1678.

Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J., and Babar, M. A. (2021). Understanding and Addressing Quality Attributes of Microservices Architecture: A Systematic Literature Review. *Inf. and Softw. Technology*, 131:106449.

Liao, H. (2024). Ensuring System Safety and Resiliency for Mission-Critical Systems During the Operations and Maintenance Phase. In *Integrated Communications, Navigation and Surveillance Conference*, pages 1–9.

Liu, X. (2024). Research on the Application of Electromechanical Automation Control Software Architecture to Production Line Inspection Under Cloud Computing Platform. In *2024 IEEE 7th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 1212–1216.

Mison, A., Davies, G., and Eden, P. (2024). Cyber Resilience, Dependability and Security. In *Int. Conf. on Cyber Warfare and Security*, pages 177–184. Academic Conferences International Limited.

Noh, G., Hui, B., and Kim, I. (2020). High Speed Train Communications in 5G: Design Elements to Mitigate the Impact of Very High Mobility. *IEEE Wireless Communications*, 27(6):98–106.

Raja, A. R., Raghunandan, K. R., Dodmane, R., Dixit, A., Kumar, M. S., and Kapila, D. (2023). Software Re-Engineering using Cloud Computing Platform. In *2023 5th Int. Conf. on Smart Systems and Inventive Technology*, pages 812–816.

Ramachandran, R. (2025). Empowering Software Architects with Artificial Intelligence: Analyzing GitHub Copilot's Role in Modern Architecture Design. In *Proc. of the 2025 7th Int. Conf. on Software Engineering and Computer Science*, pages 1–9, Taicang, China. Institute of Electrical and Electronics Engineers.

Sepúlveda, S., Cravero, A., Fonseca, G., and Antonelli, L. (2024). Systematic Review on Requirements Engineering in Quantum Computing: Insights and Future Directions. *Electronics*, 13(15).

Souza, L. S., Rocha, F. G., and Soares, M. S. (2023). A Review on Software/Systems Architecture Description for Autonomous Systems. *Recent Advances in Computer Science and Communications*, 16(3):52–60.

Steinmetz, C., Rettberg, A., Ribeiro, F. G. C., Schroeder, G., Soares, M. S., and Pereira, C. E. (2018). Using Ontology and Standard Middleware for integrating IoT based in the Industry 4.0. *IFAC-PapersOnLine*, 51(10):169–174. 3rd IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control CESCIT 2018.