A Process to Compare ATAM and Chapter 9 of ISO/IEC/IEEE 42020:2019

Gustavo S. Melo[a] and Michel S. Soares[b]

Federal University of Sergipe, Department of Computing, São Cristóvão, Brazil

Keywords: ATAM, ISO/IEC/IEEE 42020:2019, Software Architecture Evaluation.

Abstract: The evaluation of software architecture is a critical activity for ensuring system quality and alignment with business goals. The Architecture Tradeoff Analysis Method (ATAM) offers a systematic approach to identifying, prioritizing, and resolving tradeoffs within architectural decisions. In contrast, the ISO/IEC/IEEE 42020:2019 standard provides a structured framework for the design, evaluation, and documentation of software architectures in various domains. This paper presents a comparative analysis of ATAM and ISO/IEC/IEEE 42020:2019, highlighting their strengths and limitations. One conclusion is that it is important to note that the broader scope of ISO/IEC/IEEE 42020:2019 does not diminish the value of specialized methods like ATAM. Rather, it suggests a complementary relationship in which targeted evaluation techniques can be integrated into a more comprehensive framework. By examining their different approaches to architectural evaluation, this study aims to provide insights into their applicability to different contexts and implications for software architecture practices.

1 INTRODUCTION

Correct and timely decision-making for software architecture is crucial for the success of a software project (van Vliet and Tang, 2016). Architectural decisions, reviewed throughout the software lifecycle, must be based on a thorough analysis to ensure the selection of the most appropriate solutions (Chaves Costa and Soares, 2023a). Consequently, identifying quality attributes and accurately mapping usage scenarios are essential steps to avoid inefficiencies in time and resources (Agerwala and Bass, 2024).

An assessment of software architecture is a cornerstone of the development lifecycle, allowing teams to assess their planning and understanding of a project's requirements (Kaur et al., 2021). This process, conducted at various stages of the software lifecycle, helps mitigate risks, identify quality requirements, and evaluate proposed architectural solutions (Rocha et al., 2023). Assessment of software architecture anticipates potential failure points, preventing them from evolving into critical issues (Silva et al., 2023).

Proper evaluation of software architecture plays a key role in the development of robust and efficient software (Banijamali et al., 2019). Documenting and evaluating software architectures involves complex processes that require collaboration between multiple activities and stakeholders (França et al., 2017) (Santos et al., 2020). Effective communication of architectural elements such as layers, tools, decisions, components, patterns, and processes is vital to ensure that designs are understandable to different audiences (Ribeiro et al., 2017) (Kaur et al., 2021).

Several methods have been developed to analyze and validate the quality attributes of software systems. Notable among these are the Architecture Tradeoff Analysis Method (ATAM) and the ISO/IEC/IEEE 42020 standard, along with others such as SAAM, SACAM, and SBAR.

Each method provides mechanisms for documenting and assessing the architecture, ensuring alignment with project requirements, adaptability to change, and achievement of business objectives. The choice of an assessment method typically depends on factors such as the domain of the system, prioritized quality attributes, and specific characteristics of the project. In many cases, it is necessary to combine or adapt different approaches to achieve a comprehensive and effective analysis.

This paper focuses on a comparative analysis of two widely recognized approaches, ATAM and

[a] https://orcid.org/0009-0007-8141-2064
[b] https://orcid.org/0000-0002-7193-5087

Melo, G. S. and Soares, M. S.
A Process to Compare ATAM and Chapter 9 of ISO/IEC/IEEE 42020:2019.
DOI: 10.5220/0013351800003929
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 2*, pages 37-46
ISBN: 978-989-758-749-8; ISSN: 2184-4992
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

the ISO/IEC/IEEE 42020:2019 standard, highlighting their strengths, limitations, and applicability in software architecture assessment. Therefore, one can aim to provide insights into how these approaches for software evaluation support decision-making processes and contribute to architectural robustness.

2 THEORETICAL REFERENCE

ATAM (Architecture Tradeoff Analysis Method) is a structured method designed specifically for analyzing and evaluating architectural tradeoffs. The central point of the ATAM approach is to identify risks and make informed decisions throughout the software development lifecycle (Kazman et al., 2000; Müller, 2020; ISO, 2015). The idea is to refine the multiple attributes and architectural decisions according to the needs of the project, to prevent hasty observations from jeopardizing the development of the product.

ATAM has its roots in three distinct areas: the fundamental principles of architecture, the communities dedicated to analyzing quality attributes, and its direct predecessor, SAAM (Software Architecture Analysis Method). Although it shares similarities with ATAM, SAAM's main objective is to determine which architectures best meet quality requirements, taking into account the specific context of software development (Kazman et al., 1994; Müller, 2020).

There are risk assessment models that focus on unitary requirements, for example, performance, availability, and modifiability, which can skew the software product since a study focused on a single requirement can cause other requirements to be left out or go unnoticed. Using ATAM, the analysis is based on multiple requirements, making it clear which areas have trade-offs and which areas need attention for a better balance (Kazman et al., 2000; Müller, 2020).

By applying ATAM, organizations can explore in depth how architectural decisions have an impact on achieving the quality attributes that are essential to the success of the final product. ATAM contains phases defined as preparation, evaluation and consolidation of results, promoting a detailed evaluation of architectures at different stages of development, using techniques such as scenarios, allowing potential risks to be identified and mitigated before they become concrete problems (Saldana et al., 2019).

ATAM consists of nine steps:

1. **Present the ATAM.** The person responsible for the evaluation describes the evaluation method to the assembled participants, defines their expectations in line with the project's objectives, and answers the participants' questions.
2. **Present Business Drivers.** A project spokesperson describes what business goals are motivating the development efforts and hence what will be the primary architectural drivers.
3. **Present Architecture.** The architect will describe the proposed architecture through the level of detail collected during the previous two steps, focusing on how it addresses the business drivers.
4. **Identify Architectural Approaches.** Identification of the best architectural approaches by the architect and the team for the system to be developed.
5. **Generate a Quality Attribute Utility Tree.** The quality factors that comprise system "utility" (e.g., availability, security, modifiability, usability) are elicited, specified to the level of scenarios, annotated with stimuli and responses, and prioritized.
6. **Analyze Architectural Approaches.** Based on high-priority factors identified in Step 5, the architectural approaches that address those factors are elicited and analyzed. For this step, architectural risks, sensitivity points, and trade-off points are identified.
7. **Brainstorm and Prioritize Scenarios.** A larger set of scenarios is elicited from the entire group of stakeholders. This set of scenarios is prioritized through a voting process involving the entire stakeholder group.
8. **Analyze Architectural Approaches.** This step reiterates the activities of Step 6 but uses the highly ranked scenarios from Step 7. Those scenarios are considered test cases to confirm the analysis performed so far. This analysis may uncover additional architectural approaches, risks, sensitivity, and trade-off points, which are then documented.
9. **Present Results.** Based on the information collected in ATAM (which may include approaches, scenarios, attribute-specific questions, risks, the utility tree, sensitivity points, and tradeoffs), the ATAM team presents the findings to stakeholders.

The ISO/IEC/IEEE 42020:2019 standard complements processes related to ISO/IEEE 15288:2015 in the field of Systems Engineering, offering a more comprehensive approach to architecture practices. In addition, the ISO/IEC/IEEE 42020:2019 standard improves and complements the procedures established in ISO/IEC/IEEE 12207:2017 and ISO/IEC 15704:2019, effectively integrating them (ISO, 2019; ISO, 2015).

The ISO/IEC/IEEE 42020:2019 standard is intrinsically linked to ISO/IEC/IEEE 42010:2011 and ISO/IEC/IEEE 42030:2019, which are, respectively, standards for Architecture Description and Architecture Evaluation (Martin, 2018; Chaves Costa and Soares, 2023b). These related standards contribute to a consistent understanding and practice within the field of software architecture.

ISO/IEC/IEEE 42020:2019 is articulated through six main processes: Governance, Management, Conceptualization, Evaluation, Elaboration, and Enablement. Each of these processes is designed to strengthen architecture practices and promote effective implementation. Together, they form a robust framework that not only improves architectural practices but also ensures successful project development by providing a solid foundation for governance, strategy, and efficient execution.

Chapter 9 of ISO/IEC/IEEE 42020:2019 provides general guidelines for managing software architecture assessments, without delving into a specific methodology. It sets out principles and processes that can be applied in different organizational contexts and software projects. Rather than prescribing a step-by-step approach, ISO/IEC/IEEE 42020:2019 emphasizes the importance of adaptation to the specific needs of the project, collaboration between stakeholders and proper documentation of the assessment results.

Table 1 shows the functions of each process.

The Architecture Evaluation Process, as described in Chapter 9 of ISO/IEC/IEEE 42020:2019, aims to determine the suitability of one or more architectures to achieve the needs and objectives of stakeholders. Clause 9 is crucial for answering fundamental questions about the proposed architecture:

1. Is the architecture suitable for the intended uses and operational situations?
2. Is the architecture flexible and extensible enough to meet future needs?
3. Is the quality of the architecture acceptable?
4. Does the architecture address the concerns of stakeholders?
5. Does the architecture achieve its stated objectives?
6. Can the architecture be implemented successfully?

These questions are essential to ensure successful execution. The process is structured into 10 specific tasks, detailed in the standard, which guide the architecture assessment in a systematic objective way. Tasks of clause 9 are listed as follows:

Table 1: Architecture Process ISO/IEC/IEEE 42020:2019.

Process	Description
Governance	Provides direction and guidance on the architectures developed by the company and are usually decisions that come from the business level.
Management	Responsible for implementing the directives coming from governance is a process that requires continuous communication with governance, transmitting the defined strategies and progress towards achieving the objectives.
Conceptualization	Determines the appropriate solutions that must meet the needs arising from governance.
Evaluation	Determines the extent to which one or more of the architectures developed meet the needs of the stakeholders, guaranteeing linear development in line with the defined objectives.
Elaboration	Responsible for documenting the architecture(s) in a sufficiently complete manner for its intended use.
Enablement	Provides the necessary facilitators to carry out activities efficiently and intelligently.

9.1 Prepare and plan the architecture assessment effort.

9.2 Monitor, evaluate and control architecture assessment activities.

9.3 Determine evaluation objectives and criteria.

9.4 Determine evaluation methods and integrate them into the evaluation objectives and criteria.

9.5 Establish measurement techniques, methods and tools.

9.6 Collecting and reviewing evaluation-related information.

9.7 Analyze architectural concepts and properties and assess value for stakeholders.

9.8 Characterize the architecture based on the evaluation results.

9.9 Formulate conclusions and recommendations.

9.10 Capture and communicate the results of the assessment.

3 COMPARISON BETWEEN ATAM AND ISO/IEC/IEEE 42020:2019

Two widely known approaches for evaluating software architectures are ATAM and Chapter 9 of ISO/IEC/IEEE 42020:2019, each with its characteristics, similarities, and differences.

3.1 Similarities Between ATAM and ISO/IEC/IEEE 42020:2019

By correlating ATAM and ISO/IEC/IEEE 42020:2019, several similarities can be highlighted. The first is the flexibility in the order of the stages. Although numbered and sequential, both approaches allow the steps to be carried out as the architecture team and stakeholders decide to proceed. Constant communication between stakeholders and business drivers is essential for both approaches. From this communication and initial planning, the most important functional requirements, technical, economic, management constraints, key stakeholders, and business objectives are defined.

Stakeholder involvement is a feature present from the outset in both ATAM and the ISO/IEC/IEEE 42020:2019 guidelines. Stakeholders must maintain continuous communication and be involved during refinements throughout the development of the project, ensuring that their objectives are aligned and can be achieved. Both approaches emphasize the importance of continually evaluating the software architecture to ensure that it meets the quality requirements and project objectives.

Both ATAM and ISO/IEC/IEEE 42020:2019 emphasize the importance of proper documentation that contains all the objectives, requirements and possible scenarios, allowing stakeholders to have a complete understanding of the project.

ATAM and the ISO/IEC/IEEE 42020:2019 guidelines identify possible architectural approaches to be followed. In ATAM, the identification is performed in an initial phase, followed by the generation of the quality attribute tree and a thorough evaluation to correlate which architectures meet the most requirements with the best quality, according to prioritization. ISO/IEC/IEEE 42020:2019 guidelines establish techniques, methods, and tools to assist in the investigation of architectural alternatives, followed by an analysis of the architectural properties that best meet the requirements, highlighting the favourite choices.

Finally, both approaches highlight the importance of adapting to the project, with alignment meetings from the beginning of the product's conception to ensure that the main objectives are accepted by the majority of stakeholders.

3.2 Differences Between ATAM and ISO/IEC/IEEE 42020:2019

ATAM is a specific method for analyzing and evaluating architectural trade-offs, focused on identifying risks, sensitivity points, and making informed decisions during the software development process. In contrast, Chapter 9 of ISO/IEC/IEEE 42020:2019 provides general guidelines for managing architecture assessments, without delving into a specific methodology.

The ATAM evaluation scope focuses on evaluating a system's architecture in terms of quality attributes and identifying possible points of sensitivity between these attributes. ISO/IEC/IEEE 42020:2019 addresses the lifecycle management of software architecture evaluations, including the definition of evaluation criteria, the selection of appropriate evaluation methods, and the integration of evaluation results into the development process.

Although ATAM and ISO/IEC/IEEE 42020:2019 address quality attributes as an essential pillar, they do so in different ways. ATAM is highly effective in creating a comprehensive tree of quality attributes by systematically evaluating architectural decisions against multiple attributes such as performance, security, modifiability, and usability. This helps stakeholders understand the trade-offs involved in different architectural choices and their impact on system qualities. For example, ATAM can be used to determine the balance between performance and modifiability, highlighting architectural risks and their priorities. In contrast, ISO/IEC/IEEE 42020:2019 provides general guidelines for defining and managing the architecture of software-intensive systems, including guidance on architectural documentation, architectural description management, and business-oriented architecture planning.

When comparing the tasks of each method, there are some significant differences. ATAM consists of 9 tasks, while ISO/IEC/IEEE 42020:2019 defines 10 tasks. One notable difference is that ATAM's first task involves presenting the method, setting expectations, and clarifying any doubts the meeting may have. In contrast, ISO/IEC/IEEE 42020:2019 begins by noting the level of effort required for the architecture assessment, without involving a detailed initial discussion with stakeholders.

The approach to the existing architecture is another point of difference. In ATAM, after defining the business drivers, the responsible team drafts the architecture to be followed, which will be discussed and adjusted as necessary. ISO/IEC/IEEE 42020:2019, on the other hand, works directly with the identification of possible architectures that can meet the demand, without the need to write a preliminary architectural sketch.

Another distinguishing factor is brainstorming and prioritization of scenarios, which in ATAM involves bringing together all stakeholders to discuss possible scenarios that could occur. These scenarios are classified as use case scenarios and change scenarios, which can be subdivided into growth and exploratory scenarios. Growth scenarios help to identify strengths and weaknesses, while exploratory scenarios identify points of sensitivity in the software. ISO/IEC/IEEE 42020:2019 does not specify this level of detail for prioritization of scenarios.

ATAM structure is divided into three phases: preparation, evaluation, and consolidation, using techniques focused on quality attributes, scenarios and sensitivity points to conduct the evaluation. In contrast, Chapter 9 of ISO/IEC/IEEE 42020:2019 takes a more generic approach, focusing on documentation of the assessment and stakeholder collaboration throughout development.

Regarding the focus on risks, ATAM highlights risk mapping as a priority, helping the team to identify and mitigate these risks from the earliest stages of the project. ISO/IEC/IEEE 42020:2019, even tough recognizes the importance of risks, does not specifically focus on them during the architecture assessment.

4 A PROCESS TO COMPARE ATAM AND CHAPTER 9 OF ISO/IEC/IEEE 42020:2019

Based on systematic literature reviews presented before (Silva et al., 2023; Banijamali et al., 2019; Ahmadi et al., 2019; Abrahão and Insfran, 2017), the methods were analyzed to generate ideas on how to compare ATAM with the aforementioned guidelines from Chapter 9 of the ISO/IEC/IEEE 42020:2019 standard. Through the ideas collected, a process and a comparison of the evaluation criteria referred to the construction of a software architecture is carried out from its initial phase. Through this classification, it is possible to predict the sustainability (in the sense of durability) of the software architecture designed with these tools (Koziolek, 2011).

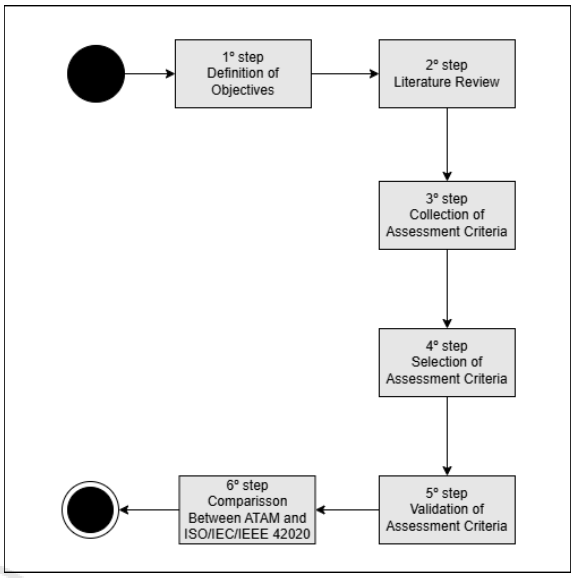
1° step
Definition of Objectives
2° step
Literature Review
3° step
Collection of Assessment Criteria
4° step
Selection of Assessment Criteria
5° step
Validation of Assessment Criteria
6° step
Comparisson Between ATAM and ISO/IEC/IEEE 42020

Figure 1: Article development process

The due diligence process in this article is summarized in six steps, as illustrated in Fig. 1.

The first step is to define the objective: to carry out a comparison between the ATAM architecture assessment methodology and the guidelines presented in Chapter 9 of ISO/IEC/IEEE 42020:2019, to identify their differences, similarities, and the evolution that has occurred over the 19 years that separate them.

The second step consists of reviewing the literature, looking for works that can guide and serve as a basis for the development of the article, also investigating evaluation criteria that can be used in the comparison.

The third step is to collect the evaluation criteria found in the literature. The fourth step is the thorough selection of tangible and intangible criteria, aiming to describe their similarities and differences.

The fifth step is the validation of the criteria to verify whether both ATAM and ISO/IEC/IEEE 42020:2019 meet the selected criteria. Finally, the sixth step is the comparison between ATAM and ISO/IEC/IEEE 42020:2019, describing the criteria presented and highlighting the improvements of ISO/IEC/IEEE 42020:2019 compared to ATAM.

Eliciting functional and non-functional requirements is the first process of Requirements Engineering. Through it, it is possible to understand and identify the needs of interested parties (Akram et al., 2024). The software architect is responsible for determining the technologies that should be used to build the software, selecting architectures appropriate to the context in which they are applied, and choosing architectural styles.

Architectural styles provide general patterns for organizing system components' and defining how they interact with each other. Choosing the right architectural style facilitates system maintenance, scalability, and adaptability, ensuring that it meets specified requirements efficiently and effectively (Power and Wirfs-Brock, 2019; Marzooni et al., 2021). These crucial points such as requirements elicitation, technology selection, choice of appropriate architectures, and definition of architectural styles are addressed in both ATAM and ISO/IEC/IEEE 42020:2019, highlighting the importance of addressing these aspects to ensure the development of software architectures that are robust and effective.

Design patterns are reusable conceptual solutions to common software design problems throughout the development cycle, providing a construction pattern that developers can follow. The application of a design pattern varies depending on the problem to be solved, making some patterns more or less suitable for certain cases (Uludağ and Matthes, 2020; Eigler et al., 2023), a point addressed by both.

The discussion and detailing of scenarios are important points covered in different ways by ATAM and ISO/IEC/IEEE 42020:2019. Both approaches involve discussing possible scenarios with stakeholders, and refining the main ideas that should be incorporated into the project. However, the detailing of scenarios is an exclusive feature of ATAM, as it is a scenario-based methodology, while ISO/IEC/IEEE 42020:2019 addresses this issue in a more generic way.

Table 2: Comparison of Requirements & Design between ATAM and ISO/IEC/IEEE 42020:2019.

Criteria	ATAM	ISO 42020
Functional Requirements	✓	✓
Non-Functional Requirements	✓	✓
Technical Selection	✓	✓
Architecture Selection	✓	✓
Architecture Style	✓	✓
Design Patterns	✓	✓
Possible Scenarios	✓	✓
Scenario Details	✓	

Adopting CI/CD (Continuous Integration and Continuous Delivery) practices plays a crucial role in new software development standards, allowing developers to reduce product release cycles and detect failures more quickly. However, despite its significant contributions, using CI/CD in practice can be laborious and generate new challenges for development teams (Zampetti et al., 2023). Monitoring and management of logs is a process that is intrinsically linked to risk mitigation. In a software architecture they are essential for monitoring and recording system behaviour, facilitating error detection, performance analysis and maintenance (Cândido et al., 2021). The selection of development tools, which will be used in conjunction with the support technologies chosen for the design of the software architecture, is a concern addressed exclusively by ISO/IEC/IEEE 42020:2019.

Table 3: Comparison of Development Tools between ATAM and ISO/IEC/IEEE 42020:2019.

Criteria	ATAM	ISO 42020
CI/CD Tools		✓
Logging Tools		✓
Versioning Tools		✓
Dependency Manager		✓
IDE Tools		✓

Two crucial topics, addressed by both ATAM and ISO/IEC/IEEE 42020:2019, are infrastructure and platform choices. These are fundamental in software development, as they directly influence efficiency, scalability, and maintenance. Selecting the appropriate infrastructure can facilitate the implementation of robust architectural standards and ensure system adaptability to changing technology and business needs. These decisions impact not only the immediate performance of software but also its evolution and ability to integrate with other technological solutions (Garcia et al., 2021).

Mapping security strategies is crucial to the success and reliability of software. Security processes, based on CIA principles (Confidentiality, Integrity, Availability), need to be discussed from the initial stages of software architecture design (Zarour et al., 2020). Software scalability is an irrefutable and necessary feature of software architectures. As the number of users of a piece of software grows, it becomes more likely that the system will be overloaded.

If the hardware used in conjunction with the software developed does not support this exponential growth, then the software could fail, revealing flaws in the design of the software architecture. Therefore, software scalability must be considered from the outset of its design to avoid future problems, maintenance costs, and even the loss of customers due to not being able to serve them (Amorim et al., 2014).

Table 4: Comparison of Infrastructure & Platform between ATAM and ISO/IEC/IEEE 42020:2019.

Criteria	ATAM	ISO 42020
Infra. Management	✓	✓
Platform Services	✓	✓

Recovery and fault tolerance strategies, also present in ATAM and ISO/IEC/IEEE 42020:2019,

Table 5: Comparison of Strategic Issues between ATAM and ISO/IEC/IEEE 42020:2019.

Criteria	ATAM	ISO 42020
Flexibility	✓	✓
Stakeholder Involvement	✓	✓
Continuous Evaluation	✓	✓
Architecture Alternatives	✓	✓
Recovery & Fault Tolerance	✓	✓
Maintenance & Support		✓
Monitoring		✓
Alignment Meetings	✓	✓
Testing Strategies	✓	✓
Communication	✓	✓
Type	Specific method	General guidelines
Evaluation Scope	Quality attributes	Lifecycle management
Quality Attributes	Tree of QA	Architecture documentation
Number of Tasks	9	10
Preliminary Architecture Sketch	✓	
General Structure	Prepare, evaluate & consolidate	Documentation & collaboration
Risk Focus	Priority	No specific focus

are crucial concerns in new software development models. High-performance software developed today seeks to have as many errors as possible due to the brutal competition imposed by the software industry. Small errors are enough for companies to suffer huge losses (Parchman et al., 2016). Maintenance and support strategies are fundamental in the development of a software architecture, as they guarantee the longevity, efficiency, and adaptability of systems. Maintenance allows software to evolve as user needs and market conditions change, fixing bugs, improving performance, and implementing new features. Support, in turn, ensures that users receive the help they need to solve problems and use the software effectively. Together, these strategies aim to achieve customer satisfaction, reduce costs in the long term and prevent failures that could compromise software execution (Hinrichs and Prifti, 2022).

Testing strategies are crucial steps in software development, by enabling exhaustive testing to identify bugs, flaws, and incorrectly implemented processes. These tests ensure software quality and reliability, increasing end-user satisfaction, and reducing costs associated with post-deployment fixes (Costa and Teixeira, 2018; Duarte et al., 2024). Effective communication and coordination strategies are necessary in software development, especially in distributed or multidisciplinary teams.

Clear and consistent communication facilitates the exchange of information, rapid problem resolution, and informed decision making, ensuring that all team members are aligned with project objectives and deadlines. Project management tools, regular meetings and detailed documentation are essential for coordinating activities, distributing responsibilities and monitoring progress. Good cooperation also helps to minimize rework and conflicts, improving efficiency and productivity (Kalogiannidis, 2020) (Muller et al., 2019). Both are concerns explained by ATAM and ISO/IEC/IEEE 42020:2019.

5 GUIDELINES FOR FUTURE WORK

For future work, we propose three possible research projects.

First, the development of a software architecture assessment framework based on the guidelines in Chapter 9 of the ISO/IEC/IEEE 42020:2019 standard. This framework would aim to:

1. Integrate the strengths of focused evaluation methods like ATAM with the comprehensive approach of ISO/IEC/IEEE 42020:2019.
2. Provide practical guidelines for implementing the standard's recommendations in real-world software development contexts.
3. Incorporate emerging trends in software architecture, such as microservices, cloud-native applications, and AI-driven systems.

Such a framework would contribute to the ongoing evolution of software architecture assessment methods, helping practitioners navigate the increasing complexity of modern software systems while adhering to international standards.

Second, research on the development and further integration of software tools that help and facilitate the use and application of architectural processes. Software tools for architecture description and evaluation are rarely employed in practice, as such tools are too primitive (simply storing and describing data) or too expensive to be used in small to medium-size companies, which are the vast majority in the software industry (da Costa Junior et al., 2019). In addition, the inclusion of emerging technologies, such as techniques of machine learning with the aforementioned tools to increase efficiency and effectiveness in decision-making in the field of software architecture, as well as the continued exploration of best practices and evolving standards in architectural frameworks and modelling languages, which remain key to keeping up with advances in the industry.

Third, develop a standardized set of metrics and automation mechanisms for architecture evaluation methodologies. This research direction would focus on:

1. Creating quantifiable metrics to measure the effectiveness of different architecture evaluation methods in various contexts.
2. Developing automated tools for collecting and analyzing these metrics during the evaluation process.
3. Establishing benchmarks and comparative analyses between different evaluation methodologies based on empirical data.
4. Investigating the correlation between architecture evaluation results and actual project outcomes.

This research would help organizations make more informed decisions about which evaluation methodology to use based on concrete data, while also providing a foundation for continuous improvement of these methodologies through automated feedback and measurement systems. The automation aspect would address the current challenge of manual-intensive evaluation processes, making architecture evaluation more accessible and efficient for organizations of all sizes.

6 CONCLUSION

This comparative study between ATAM and ISO/IEC/IEEE 42020:2019 reveals significant advancements in software architecture evaluation methodologies over the past two decades. ATAM continues to be a valuable tool in the software industry, the emergence of ISO/IEC/IEEE 42020:2019 represents a more comprehensive approach to software architecture development and assessment.

The analysis highlights several key improvements embodied in ISO/IEC/IEEE 42020:2019:

1. **Broader Scope.** The standard encompasses a wider range of aspects in software architecture development, extending beyond evaluation to include crucial elements such as tool selection and development processes.
2. **Integration of Modern Practices.** ISO/IEC/IEEE 42020:2019 incorporates contemporary software development practices, including continuous integration and delivery (CI/CD) and systematic logging for failure analysis.
3. **Emphasis on Maintenance and Monitoring.** The standard places greater emphasis on long-term aspects of software architecture, including maintenance strategies and monitoring practices.
4. **Comprehensive Task Framework.** ISO/IEC/IEEE 42020:2019 outlines a more extensive set of tasks for the design and evaluation of software architectures, reflecting the increased complexity of modern software systems.
5. **Enhanced Stakeholder Collaboration.** The standard emphasizes the importance of documentation and collaboration with stakeholders throughout the architecture development process.

These advancements reflect the Software Engineering community, evolving understanding of the complexities involved in designing and evaluating software architectures. The shift from ATAM's focused approach on quality attribute tradeoffs to ISO/IEC/IEEE 42020:2019's holistic view of architecture development demonstrates the field's maturity.

However, it is important to note that the broader scope of ISO/IEC/IEEE 42020:2019 does not diminish the value of specialized methods such as ATAM. Rather, it suggests a complementary relationship where targeted evaluation techniques can be integrated into a more comprehensive framework.

REFERENCES

Abrahão, S. and Insfran, E. (2017). Evaluating Software Architecture Evaluation Methods: An Internal Replication. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, EASE '17, pages 144—-153, New York, NY, USA. Association for Computing Machinery.

Agerwala, G. and Bass, L. (2024). Teaching Software Architecture Design - Building Intuition. In *Proceedings of the 1st International Workshop on Designing Software*, Designing '24, pages 27—-33, New York, NY, USA. Association for Computing Machinery.

Ahmadi, H., Farahani, B., Aliee, F. S., and Motlagh, M. A. (2019). Cross-layer Enterprise Architecture Evaluation: An Approach to Improve the Evaluation of TO-BE Enterprise Architecture. In *Proceedings of the*

International Conference on Omni-Layer Intelligent Systems, COINS '19, pages 223—228, New York, NY, USA. Association for Computing Machinery.

Akram, F., Ahmad, T., and Sadiq, M. (2024). Recommendation Systems-based Software Requirements Elicitation Process—a Systematic Literature Review. *Journal of Engineering and Applied Science*, 71.

Amorim, S. d. S., Almeida, E. S. d., and McGregor, J. D. (2014). Scalability of Ecosystem Architectures. In *2014 IEEE/IFIP Conference on Software Architecture*, pages 49–52.

Banijamali, A., Heisig, P., Kristan, J., Kuvaja, P., and Oivo, M. (2019). Software Architecture Design of Cloud Platforms in Automotive Domain: An Online Survey. In *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, pages 168–175.

Chaves Costa, J. E. and Soares, M. S. (2023a). A Review on the Capacities of the ISO/IEC/IEEE 42020:2019 Standard for Architecture Elaboration of Software and Systems. In *Proceedings of the XIX Brazilian Symposium on Information Systems*, SBSI '23, pages 412––418, New York, NY, USA. Association for Computing Machinery.

Chaves Costa, J. E. and Soares, M. S. S. (2023b). A Review on the Capacities of the ISO/IEC/IEEE 42020:2019 Standard for Architecture Elaboration of Software and Systems. In *Proceedings of the XIX Brazilian Symposium on Information Systems*, pages 412—-418, New York, NY, USA. Association for Computing Machinery.

Costa, A. and Teixeira, L. (2018). Testing Strategies for Smart Cities applications: A Systematic Mapping Study. In *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing*, SAST '18, pages 20—-28, New York, NY, USA. Association for Computing Machinery.

Cândido, J., Aniche, M., and van Deursen, A. (2021). Log-based Software Monitoring: A Systematic Mapping Study. *PeerJ Computer Science*, 7:1–38.

da Costa Junior, A. A., 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. N., Korkhov, V., Torre, C. M., Rocha, A. M. A. C., Taniar, D., Apduhan, B. O., and Tarantino, E., editors, *Computational Science and Its Applications - ICCSA 2019 - 19th International Conference, Saint Petersburg, Russia, July 1-4, 2019, Proceedings, Part V*, volume 11623 of *Lecture Notes in Computer Science*, pages 31–42. Springer.

Duarte, Y., Durelli, V., Nardi, P. A., and Endo, A. T. (2024). Exploratory Testing Strategies for Video Games: An Experience Report. In *Proceedings of the 22nd Brazilian Symposium on Games and Digital Entertainment*, SBGames '23, pages 46—-55, New York, NY, USA. Association for Computing Machinery.

Eigler, T., Huber, F., and Hagel, G. (2023). Tool-Based Software Engineering Education for Software Design Patterns and Software Architecture Patterns - a Systematic Literature Review. In *Proceedings of the 5th European Conference on Software Engineering Education*, ECSEE '23, pages 153—-161, New York, NY, USA. Association for Computing Machinery.

França, J. M. S., de S. Lima, J., and Soares, M. S. (2017). Development of an Electronic Health Record Application using a Multiple View Service Oriented Architecture. In Hammoudi, S., Smialek, M., Camp, O., and Filipe, J., editors, *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 2, Porto, Portugal, April 26-29, 2017*, pages 308–315. SciTePress.

Garcia, J., Mirakhorli, M., Xiao, L., Zhao, Y., Mujhid, I., Pham, K., Okutan, A., Malek, S., Kazman, R., Cai, Y., and Medvidović, N. (2021). Constructing a Shared Infrastructure for Software Architecture Analysis and Maintenance. In *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, pages 150–161.

Hinrichs, M. and Prifti, L. (2022). Visualizing Maintenance Data to Support Decisions on Strategic Maintenance Planning. In *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '22, pages 473—479, New York, NY, USA. Association for Computing Machinery.

ISO (2015). Systems and Software Engineering — System Life Cycle Processes.

ISO (2019). Enterprise, Systems and Software — Architecture Processes. Final Draft International Standard (FDIS).

Kalogiannidis, S. (2020). Impact of Effective Business Communication on Employee Performance. *European Journal of Business and Management Research*, 5(6).

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 International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 1673–1678.

Kazman, R., Bass, L., Abowd, G., and Webb, M. (1994). SAAM: A Method for Analyzing the Properties of Software Architectures. In *Proceedings of 16th International Conference on Software Engineering*, pages 81–90. IEEE.

Kazman, R., Klein, M., Clements, P., Northrop, L., and McGregor, J. (2000). ATAM: Method for Architecture Evaluation. In *Proceedings of the 22nd international conference on Software engineering*, pages 478–487. ACM.

Koziolek, H. (2011). Sustainability Evaluation of Software Architectures: A Systematic Review. In *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*, QoSA-ISARCS '11, pages 3––12, New York, NY, USA. Association for Computing Machinery.

Martin, J. N. (2018). Overview of an Emerging Standard on Architecture Processes — ISO/IEC/IEEE 42020. In

2018 Annual IEEE International Systems Conference (SysCon), pages 1–8, Vancouver, BC, Canada. IEEE.

Marzooni, H. H., Motameni, H., and Ebrahimnejad, A. (2021). Architecture Style Selection using Statistics of Quality Attributes to Reduce Production Costs. *International Arab Journal Of Information Technology*, 18(4):513–522.

Müller, H. (2020). Software Architecture Evaluation Methods and Tools: Analyzing Methods and Tools for Evaluating Software Architectures to Ensure Adherence to Quality Attributes and Design Principles. *Distributed Learning and Broad Applications in Scientific Research*, 6:1–14.

Muller, M., Fussell, S. R., Gao, G., Hinds, P. J., Oliveira, N., Reinecke, K., Robert, L., Siangliulue, K. P., Wulf, V., and Yuan, C.-W. (2019). Learning from Team and Group Diversity: Nurturing and Benefiting from our Heterogeneity. In *Companion Publication of the 2019 Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '19 Companion, page 498–505, New York, NY, USA. Association for Computing Machinery.

Parchman, Z. W., Vallee, G. R., Naughton, T., Engelmann, C., Bernholdt, D., and Scott, S. L. (2016). Adding Fault Tolerance to NPB Benchmarks Using ULFM. In *Proceedings of the ACM Workshop on Fault-Tolerance for HPC at Extreme Scale*, FTXS '16, pages 27—-34, New York, NY, USA. Association for Computing Machinery.

Power, K. and Wirfs-Brock, R. (2019). An Exploratory Study of Naturalistic Decision Making in Complex Software Architecture Environments. In Bures, T., Duchien, L., and Inverardi, P., editors, *Software Architecture, ECSA 2019*, volume 11681 of *Lecture Notes in Computer Science*, pages 55–70. Univ Lille; I Site ULNE; Inria; CNRS, Miss Femmes; CNRS, GDR, Genie Programmation Logiciel; CNRS, UMR, CRIStAL Comp Sci Lab; Spirals Res Grp. 13th European Conference on Software Architecture Engineering (ECSA), Paris, FRANCE, SEP 09-13, 2019.

Ribeiro, Q. A. D. S., Ribeiro, F. G. C., and Soares, M. S. (2017). A Technique to Architect Real-time Embedded Systems with SysML and UML through Multiple Views. In Hammoudi, S., Smialek, M., Camp, O., and Filipe, J., editors, *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 2, Porto, Portugal, April 26-29, 2017*, pages 287–294. SciTePress.

Rocha, F. G., Misra, S., and Soares, M. S. (2023). Guidelines for Future Agile Methodologies and Architecture Reconciliation for Software-Intensive Systems. *Electronics*, 12(7).

Saldana, Y. P., Toribio, G. G., Hernandez, M. J. S., Mora, J. J. H., Bautista, H. N., Alegria, J. A. H., Ordonez, C. A. C., Davila, L. M., and Gutierrez, N. A. (2019). Evaluation of the Modifiability of an Evolution System Using the ATAM Method. *International Journal of Science and Research (IJSR)*, 8(2):1772–1779.

Santos, V. M., Misra, S., and Soares, M. S. (2020). Architecture Conceptualization for Health Information Systems Using ISO/IEC/IEEE 42020. In Gervasi, O., Murgante, B., Misra, S., Garau, C., Blecic, I., Taniar, D., Apduhan, B. O., Rocha, A. M. A. C., Tarantino, E., Torre, C. M., and Karaca, Y., editors, *Computational Science and Its Applications - ICCSA 2020 - 20th International Conference, Cagliari, Italy, July 1-4, 2020, Proceedings, Part VI*, volume 12254 of *Lecture Notes in Computer Science*, pages 398–411. Springer.

Silva, S., Tuyishime, A., Santilli, T., Pelliccione, P., and Iovino, L. (2023). Quality Metrics in Software Architecture. In *2023 IEEE 20th International Conference on Software Architecture (ICSA)*, pages 58–69.

Uludağ, O. and Matthes, F. (2020). Large-Scale Agile Development Patterns for Enterprise and Solution Architects. In *Proceedings of the European Conference on Pattern Languages of Programs 2020*, EuroPLoP '20, New York, NY, USA. Association for Computing Machinery.

van Vliet, H. and Tang, A. (2016). Decision Making in Software Architecture. *Journal of Systems and Software*, 117:638–644.

Zampetti, F., Tamburri, D., Panichella, S., Panichella, A., Canfora, G., and Di Penta, M. (2023). Continuous Integration and Delivery Practices for Cyber-Physical Systems: An Interview-Based Study. *ACM Trans. Softw. Eng. Methodol.*, 32(3).

Zarour, M., Alenezi, M., and Alsarayrah, K. (2020). Software Security Specifications and Design: How Software Engineers and Practitioners Are Mixing Things Up. In *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, EASE '20, pages 451—456, New York, NY, USA. Association for Computing Machinery.