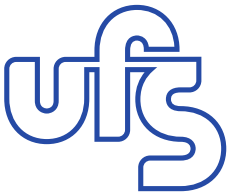


FEDERAL UNIVERSITY OF SERGIPE

CENTER OF EXACT SCIENCES AND TECHNOLOGY

GRADUATE PROGRAM IN COMPUTER SCIENCE

A proposal for Software Architecture Elaboration guided by Clause 10 of the ISO/IEC/IEEE 42020 Standard

Master Thesis

José Eduardo Chaves Costa

São Cristóvão – Sergipe

2024

FEDERAL UNIVERSITY OF SERGIPE

CENTER OF EXACT SCIENCES AND TECHNOLOGY

GRADUATE PROGRAM IN COMPUTER SCIENCE

José Eduardo Chaves Costa

A proposal for Software Architecture Elaboration guided by Clause 10 of the ISO/IEC/IEEE 42020 Standard

A Master's Thesis submitted to the Graduate Program in Computer Science at the Federal University of Sergipe in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Advisor: Dr. Michel dos Santos Soares

São Cristóvão – Sergipe

2024

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Costa, José Eduardo Chaves
C837p A proposal for software architecture elaboration guided by clause 10 of the ISO/IEC/IEEE 42020 standard / José Eduardo Chaves Costa ; orientador Michel dos Santos Soares. - São Cristóvão, 2024.
118 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2024.

1. Arquitetura de software. 2. Processamento eletrônico de dados - Documentação. 3. Computação. I. Soares, Michel dos Santos orient. II. Título.

CDU 004.27

UNIVERSIDADE FEDERAL DE SERGIPE
PRO-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS. Candidato: José Eduardo Chaves Costa

Em 26 dias do mês de julho do ano de dois mil e vinte quatro, com início às 14hs, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **José Eduardo Chaves Costa**, que desenvolveu o trabalho intitulado: **"A proposal for Software Architecture Elaboration guided by Clause 10 of the ISO/IEC/IEEE 42020 Standard",** sob a orientação do Prof. Dr. **Michel dos Santos Soares.** A Sessão foi presidida pelo Prof. Dr. **Michel dos Santos Soares** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, o Dr. **Valdemar Vicente Graciano Neto (**UFG**)** e, em seguida, Dr. **Fábio Gomes Rocha** (PROCC/UFS) . Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) **Aprovado** "*(aprovado/reprovado)"*. Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 26 de julho de 2024.

gov.br Documento assinado digitalmente
MICHEL DOS SANTOS SOARES
Data: 01/08/2024 09:02:50-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Michel dos Santos Soares
(PROCC/UFS)
Presidente

gov.br Documento assinado digitalmente
VALDEMAR VICENTE GRACIANO NETO
Data: 26/07/2024 15:50:49-0300
Verifique em https://validar.iti.gov.br

Prof. Dr Valdemar Vicente Graciano Neto
(UFG)
Examinador Externo

gov.br Documento assinado digitalmente
FABIO GOMES ROCHA
Data: 26/07/2024 16:02:16-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Fábio Gomes Rocha
(PROCC/UFS)
Examinador Interno

gov.br Documento assinado digitalmente
JOSE EDUARDO CHAVES COSTA
Data: 31/07/2024 11:05:23-0300
Verifique em https://validar.iti.gov.br

José Eduardo Chaves Costa
Candidato

I dedicate this work to my family, for their constant love, support, and encouragement throughout this journey. To my advisor, for your invaluable guidance and mentorship.

Acknowledgements

The completion of this dissertation would not have been possible without the support and guidance of various individuals and institutions. I would like to express my deepest gratitude to everyone who contributed in any way to the realization of this work.

I would like to express my gratitude to my advisor, Prof. Dr. Michel dos Santos Soares, for his patience, guidance, and support throughout this endeavor. His valuable suggestions and insights were instrumental in the advancement of this research.

I would like to express my gratitude to the members of the examining board, Prof. Dr. Fabio Gomes Rocha and Prof. Dr. Valdemar Vicente Graciano Neto, for accepting the invitation to serve on the board.

I would like to express my gratitude to the Federal University of Sergipe for granting me leave for qualification, which has allowed me to dedicate myself fully to this research.

I would like to express my gratitude to my parents and family, particularly to my wife Aline and daughters Maria Eduarda and Esmeralda, for their unwavering love, emotional support, and encouragement throughout this journey. This accomplishment would not have been attained without your contributions.

Finally, I would like to thank all the institutions and people who, directly or indirectly, contributed to this work. My sincere thanks to everyone.

Resumo

Este trabalho de mestrado apresenta um estudo abrangente sobre a elaboração de arquitetura de software orientada pela Cláusula 10 da norma ISO/IEC/IEEE 42020. A crescente complexidade dos sistemas de software exige novas metodologias para modelagem e documentação da arquitetura de software, visando aprimorar os processos de tomada de decisão entre as partes interessadas. A pesquisa destaca os desafios enfrentados na documentação da arquitetura de software, incluindo baixa integração, descrições informais e falta de coerência, o que pode impedir a comunicação e o entendimento eficazes entre desenvolvedores, arquitetos e engenheiros.

O trabalho inicia explorando a evolução da documentação de arquitetura de software, enfatizando seu papel crítico no gerenciamento da complexidade e facilitação da comunicação entre as partes interessadas. Ele analisa metodologias e frameworks existentes, como Views and Beyond (V&B) e Rozanski & Woods (R&W), destacando suas contribuições para a estruturação dos processos de arquitetura de software. Em seguida, o estudo apresenta a norma ISO/IEC/IEEE 42020, detalhando seus seis processos — Governança de Arquitetura, Gerenciamento, Conceituação, Avaliação, Elaboração e Habilitação — cada um voltado para o aprimoramento do desenvolvimento e gerenciamento de arquiteturas de software.

A principal contribuição deste trabalho é o desenvolvimento de um processo para a elaboração da arquitetura de software, especificamente focado na Cláusula 10 da norma ISO/IEC/IEEE 42020. Esse processo aborda as complexidades e desafios da documentação arquitetural, visando simplificar e padronizar atividades e tarefas. Ele oferece uma abordagem estruturada que visa facilitar um melhor entendimento e implementação das práticas de arquitetura de software. A pesquisa emprega a metodologia Design Science Research (DSR), garantindo um desenvolvimento rigoroso e sistemático do processo proposto.

Uma importante contribuição prática deste estudo é o desenvolvimento do ArchCE, uma ferramenta de software projetada para apoiar o processo desenvolvido. O ArchCE auxilia *stakeholders* no gerenciamento e documentação de arquiteturas de software, fornecendo uma aplicação prática do arcabouço teórico. Ele agiliza o processo de documentação, garantindo o alinhamento com os requisitos e preocupações dos *stakeholders*. A ferramenta se destina a ser disponibilizada publicamente, promovendo transparência, colaboração e envolvimento da comunidade em seu desenvolvimento e aprimoramento contínuos.

Em conclusão, este trabalho de mestrado contribui substancialmente para o campo da arquitetura de software ao preencher a lacuna entre os padrões teóricos e a implementação prática. Ele oferece uma abordagem abrangente e padronizada que auxilia os profissionais no gerenciamento e entendimento eficazes dos processos de arquitetura de software. O desenvolvimento da ferramenta

ArchCE aumenta ainda mais a aplicabilidade da pesquisa, fornecendo mais um recurso para todos os envolvidos no desenvolvimento de arquiteturas de software. Esta pesquisa apoia o avanço das melhores práticas em documentação de arquitetura de software, contribuindo, em última instância, para o desenvolvimento de sistemas de software mais eficientes e eficazes.

Palavras-chave: Arquitetura de software, ISO/IEC/IEEE 42020, Elaboração de arquitetura, Documentação, Padronização de processos.

Abstract

This master's thesis presents a comprehensive study on software architecture elaboration guided by Clause 10 of the ISO/IEC/IEEE 42020 standard. The increasing complexity of software systems necessitates new methodologies for modeling and documenting software architecture, aiming to improve decision-making processes among stakeholders. The research highlights the challenges faced in software architecture documentation, including poor integration, informal descriptions, and lack of coherence, which can impede effective communication and understanding among developers, architects, and engineers.

The thesis begins by exploring the evolution of software architecture documentation, emphasizing its critical role in managing complexity and facilitating stakeholder communication. It reviews existing methodologies and frameworks, such as Views and Beyond (V&B) and Rozanski & Woods (R&W), underscoring their contributions to structuring software architecture processes. The study then introduces the ISO/IEC/IEEE 42020 standard, detailing its six processes – Architecture Governance, Management, Conceptualization, Evaluation, Elaboration, and Enablements – each aimed at enhancing the development and management of software architectures.

The core contribution of this thesis is the development of a process for software architecture elaboration, specifically focused on Clause 10 of the ISO/IEC/IEEE 42020 standard. This process addresses the complexities and challenges in architectural documentation, aiming to simplify and standardize activities and tasks. It offers a clear, structured approach to facilitate a better understanding and implementation of software architecture practices. The research employs the Design Science Research (DSR) methodology, ensuring a rigorous and systematic development of the proposed process.

A significant practical contribution of this study is the development of ArchCE, a software tool designed to support the developed process. ArchCE aids stakeholders in managing and documenting software architectures, providing a practical application of the theoretical concepts. It streamlines the documentation process, ensuring alignment with stakeholder requirements and concerns. The tool is intended to be publicly available, promoting transparency, collaboration, and community involvement in its ongoing development and improvement.

In conclusion, this thesis makes a substantial contribution to the field of software architecture by bridging the gap between theoretical standards and practical implementation. It offers a comprehensive, standardized approach that aids professionals in effectively managing and understanding software architecture processes. The development of the ArchCE tool further enhances the applicability of the research, providing a valuable resource for stakeholders involved in software architecture development. By promoting clarity, consistency, and collaboration, this

research supports the advancement of best practices in software architecture documentation, ultimately contributing to the development of more efficient and effective software systems.

Keywords: Software architecture, ISO/IEC/IEEE 42020, Architecture elaboration, Documentation, Process standardization

List of Figures

List of Tables

Contents

1 Introduction

The increasing complexity of software systems leads to a growing interest in new methodologies for modeling and communicating software architectures. Since the 1990s, authors such as Garlan and Shaw (1993) and Abowd, Allen and Garlan (1995) have discussed the challenges of modeling and communicating software systems in their entirety, namely, the description of the system's structure or its architecture. In their articles, those authors addressed organizational structural issues as well as descriptions of software components, their interconnections, quality attributes, architectural views and viewpoints, and architectural styles as complex design decisions.

When developers, architects, and software engineers discuss or present high-level documentation, they are making and communicating design decisions to stakeholders involved in the development and use of software architectures. These decisions are recorded in architectural system models, which explain and justify the design choices. Architectural system models can be described in various ways, such as informal diagrams, descriptive terms, specific domain models and frameworks, practical rules, or idiomatic patterns that emerge informally, as described by Garlan and Shaw (1993), Abowd, Allen and Garlan (1995), Hasselbring (2018), and Ozkaya (2018).

To unify practices developed by various organizations or individuals, the ISO (International Organization for Standardization), the IEC (International Electrotechnical Commission), and the IEEE (Institute of Electrical and Electronics Engineers), organizations specialized in global standardization, have published a series of ISO/IEC/IEEE 420×0 standards (such as ISO/IEC/IEEE 42020, ISO/IEC/IEEE 42030, and ISO/IEC/IEEE 42010) that have significantly influenced software architecture development practices (KUMAR; LOKKU; NATARAJAN, 2021a). These standards establish processes and provide a common vocabulary with best practices for describing, governing, managing, synthesizing, and evaluating architectures throughout the software development life cycle (KUMAR; LOKKU; NATARAJAN, 2021a).

Among these standards, ISO/IEC/IEEE 42020 (henceforth, 42020) presents the Clause

10 named Architecture Elaboration, which offers a standardized and formal way of documenting software architectures. Clause 10 is one of the six clauses described in the 42020 standard titled Software, systems and enterprise — Architecture processes. Published in 2019, this standard complements and encompasses architectural practices of the Architecture Definition process found in international standards ISO/IEC/IEEE 12207 (henceforth, 12207) and ISO/IEC/IEEE 15288 (henceforth, 15288) (MARTIN, 2015). The Architecture Elaboration process presents activities and tasks for precisely modeling and communicating architectural viewpoints, views, and models for intended objectives, ensuring alignment with stakeholders' requirements, concerns, and constraints (IEEE, 2019).

The objective of this study is to create a process that simplifies the activities and tasks of the aforementioned clause, taking the Architecture Elaboration clause (Clause 10) and the 42020 standard as analytical objects. The process aims to reduce difficulties in understanding concepts, properties, and principles related to software architecture development. It is hoped that the process will contribute to a better understanding and application of software architecture development standards, assisting professionals and organizations involved in the context of developing software architectures.

1.1 Research Problem

The iterative process of elaborating software architectures entails the continuous generation of diverse documents, often characterized by poor integration with informal descriptions and lack of coherence among themselves as perceived by software developers (LAND, 2002; FAIRBANKS, 2010; COSTA; SOARES, 2023). Furthermore, standards can also pose challenges in understanding and practical application. The complexity of standards may hinder reading and writing of software architecture documentation, which is common among developers, architects, software engineers, and others involved in software architecture development and usage (SOARES, 2017).

Studies in the field show that these professionals may encounter difficulties in understanding architectural concepts, properties, and principles, even when they are aware of the usage context (RIBEIRO; RIBEIRO; SOARES, 2017; SOARES; VRANCKEN; WANG, 2010; KUMAR; LOKKU; NATARAJAN, 2021a). As a result, it is common to observe that these professionals or organizations dealing with software systems may reduce their activities, resorting to improved ways of describing architecture or even describing it without any rigor.

Such misunderstandings are understandable given the complexity and scale inherent in software systems, rendering them structurally vast and challenging to communicate effectively (FAIRBANKS, 2010). Despite these formidable challenges, developers exhibit remarkable resilience in confronting adversity, grounded in the ongoing evolution of development methodologies, architectural paradigms, and tooling.

Despite various evolutions, the continuous refinement of software architecture documentation reflects the adaptive nature of the development landscape (FAIRBANKS, 2010). Considering the symbiotic relationship between comprehensive documentation and successful software architecture, developers play a crucial role in nurturing an environment where understanding thrives, challenges are met with informed solutions, and methodological evolution endures (CLEMENTS et al., 2010).

In this specific context, resilience and innovation intersect to tackle the intricate facets of software systems. Clause 10, identified as the Architecture Elaboration process, within the 42020, proposes a way for articulating software architecture by offering detailed and precise descriptions to describe or document software architecture for intended purposes (IEEE, 2019). Through this process, all stakeholders involved in software architecture elaboration gain access to a robust method that serves as a "process reference model" as cited in IEEE (2019), allowing its adaptation to create other methods or processes to address the complexity of documenting software.

However, to portray software architecture documentation as a definitive solution requires a nuanced perspective. The absence of flawless solutions or silver bullets," a notion already elucidated by Brooks (1987), is acknowledged in this thesis. Even though software architecture documentation is not a definitive solution, it serves crucial functions, as explained by Fairbanks (2010): ... It aids in software partitioning, imparts knowledge facilitating better software design, and furnishes abstractions aiding in software comprehension." To put it simply, it facilitates the partitioning of software, enhances expertise in software design, and provides abstractions for analyzing software.

It is essential to emphasize the importance of a well-designed software architecture, as it ensures system quality by satisfying and reinforcing critical quality attributes. When development is conducted in an ad hoc manner, there is a significant risk of neglecting these attributes, which can compromise code quality and result in rework. Therefore, it is crucial to propose a structured process to mitigate the risks associated with ad hoc development, highlighting the importance of careful process design to ensure the robustness and quality of the system.

In practical terms, the endeavor to create a robust architecture demands parallel attention to detailed and unambiguous documentation (BASS; CLEMENTS; KAZMAN, 2021a). The documentation assumes the role of the architect's articulate voice, facilitating comprehension, utilization, and future maintenance (FAIRBANKS, 2010). Its importance goes beyond the project phase and keeps the architecture valuable even after the original developers have moved on. In essence, documentation is an indispensable component, not only for representing the architect in the present but also for guiding future project stewardship.

To address the challenges of software architecture documentation, a systematic approach for describing a process for elaborating software architecture documentation under the Architecture Elaboration process, as presented in Chapter 4, is proposed. The objective is to utilize the provided

framework as a foundation for developing a customized and comprehensive process. The resulting process model will aid in the conceptualization and organization of the intricate aspects inherent in the elaboration of software or system architecture.

1.2 Background and Related Work

An exploration of foundational concepts and existing research pertinent to software architecture documentation is presented in this section. It commences with an analysis of the evolution and significance of software architecture documentation, emphasizing its role in managing complexity and facilitating communication among stakeholders. Thereafter, it explores various methodologies and frameworks utilized in software architecture, emphasizing their contributions to the structuring and organization of software architecture documentation processes. Such a background provides the foundation for a more detailed analysis of current practices and challenges in software architecture documentation.

1.2.1 Software Architecture Documentation

Software systems have been experiencing unprecedented growth in both size and complexity for decades (GARLAN; SHAW, 1993; FAIRBANKS, 2010; BASS; CLEMENTS; KAZMAN, 2021b). As software systems expand, the focus shifts from algorithm and data structure selection to encompassing component organization, control structures, communication protocols, data access, design elements, physical distribution, and design alternatives, all constituting the domain of software architecture (GARLAN; SHAW, 1993).

Software architecture orchestrates the design and overall system structure to ensure seamless integration of component sets, fostering a cohesive and successful entirety (CLEMENTS et al., 2010). It transforms the abstract concept of software architecture into concrete documentation, guiding architects in design and structural matters, as well as instructing developers in precise execution and decision-making. Furthermore, it serves as a record of these decisions, providing valuable insights into the architect's strategic solutions for prospective custodians of the system.

Originating from the inherently abstract domain of software architecture, it elucidates methods to coordinate design and system structure for the coherent integration of component sets (CLEMENTS et al., 2010). It further provides essential guidance to architects and developers, assists newcomers in understanding both existing and new projects, facilitates communication among stakeholders, and documents crucial decisions for future reference (ERNST; ROBILLARD, 2023). These functions are crucial for effective project management.

Despite the meticulous creation of software architecture, if it is not properly understood, or if it is implemented incorrectly by its developers, the outcome will invariably be suboptimal (CLEMENTS et al., 2010). In the domain of software architecture, developers are continuously involved in decision-making, spanning both major and minor considerations. However, it is

common for developers to forego formal documentation of these decisions, opting instead to implicitly integrate them into the source code or software design models (TIAN; LIANG; BABAR, 2022). These models might include use case models, which delineate decisions regarding the interactions among system components and external entities, or component models that encapsulate structural decisions, such as pattern selection. Alternatively, physical models might be employed for decisions related to deployment, such as the selection of runtime environments. Nonetheless, stakeholders, including developers, clients, and other architects, may lack the time or inclination to peruse various architectural perspectives. Clear design guidance is sought by developers, clients require comprehension of environmental shifts to align with business objectives, and other architects seek a succinct understanding of critical aspects and the original architect's considerations. Thus, explicit documentation is indispensable for delivering decisive guidance and clarity on architectural decisions and facilitating knowledge management within software architecture (CAPILLA et al., 2016; KALSI, 2024).

1.2.2 Processes for documenting software architecture

At this juncture, it is assumed that the reader has a foundational comprehension of the significance of documentation. The subsequent discourse will delve into the rationale behind establishing a documentation process and the existing insights regarding the software architecture description or documentation process.

The documentation of software architecture requires the intricate coordination of activities, the engagement of diverse stakeholders, and the proficiency in capturing software architecture knowledge. This endeavor is pivotal, as it furnishes a framework for orchestrating and supervising these multifaceted activities. The process aims to impose structure amidst chaos by organizing activities and tasks, thereby transforming inputs (e.g., needs, requirements, concerns) into outputs (e.g., architecture description or documentation) (CAPILLA et al., 2016; IEEE, 2017; KNEUPER, 2018)..

The foremost methods for structuring and organizing software architecture documentation are predominantly grounded in the concept of architectural views (KRUCHTEN, 1995) (ERNST; ROBILLARD, 2023). Among these, two notable frameworks emerge, albeit not as processes but as frameworks. The Views and Beyond (V&B) (CLEMENTS et al., 2010), and the methodology (R&W) (ROZANSKI; WOODS, 2012) based on ISO/IEC/IEEE 42010 (henceforth, 42010), stand out as prominent frameworks for documenting software architecture.

The V&B framework underscores a structured and systematic approach to architectural documentation. It entails identifying and selecting appropriate views, crafting documentation for each view, and ensuring coherence across views. The practical framework furnished in the literature assists architects in devising effective documentation conducive to communication and decision-making across the software development lifecycle.

Conversely, the R&W methodology directs attention towards stakeholders and their viewpoints. In the R&W methodology, the viewpoints of stakeholders, or their perspectives on the system architecture, are of paramount importance. The focus is on comprehending the diverse perspectives of stakeholders and tailoring documentation to address their specific concerns and interests. The objective of this approach is to ensure that the software architecture documentation remains pertinent and beneficial to all stakeholders engaged in the project.

These approaches, along with DODAF, TOGAF, UAF, Zachman Framework, and other well-supported frameworks, are characterized by systematic approaches, which contrast to numerous examples of documentation that do not adhere to a predetermined format (ERNST; ROBILLARD, 2023). Furthermore, it is noteworthy that a limited number of methodologies offer a practical, step-by-step guide, articulating the activities and tasks to be executed within design iterations, particularly in the context of 42020.

In Hofmeister et al. (2007), a general model for software architecture design is proposed, characterizing a comprehensive process. The first activity in this process is architectural analysis, which aims to define the problems that the architecture needs to address by formulating the significant architectural requirements, known as ASRs. The second activity is architectural synthesis, which results in candidate architectural solutions that meet the identified ASRs. The third activity, architectural evaluation, ensures that the architectural decisions made are the most appropriate. The model also incorporates the concept of a backlog, which connects the various activities and emphasizes the importance of incremental and continuous evaluation throughout the entire design process.

In Cervantes and Kazman (2016), a practical approach to designing software architectures named Attribute-Driven Design 3.0 is presented, describing a seven-step process that underscores the importance of clear inputs, prioritization, and iterative refinement to achieve the desired architectural documentation. In Martin and O'Neil (2021), a nine-step process is proposed for creating enterprise architecture views aligned with the Unified Architecture Framework standard, catering to stakeholder domains.

In the final section, a significant connection is drawn with the conceptualization, evaluation, and elaboration processes outlined in 42020 for software architecture description. However, it is noted that there is no explicit adoption of the more granular and adaptable approach to the mentioned processes, especially the software architecture elaboration process. The authors correlate aspects with the Unified Architecture Framework, but do not directly employ the processes of 42020.

The three most recent studies mentioned present a practical approach and are characterized as processes, according to the definition established by Kneuper (2018). However, the main difference compared to the process described in this dissertation lies in the greater granularity: a process was developed with a larger number of steps, each accompanied by a detailed definition.

1.3 Objectives

The objective of this study is to develop a process that defines the activities, tasks, methodologies, and techniques (the "why" and "how") necessary to achieve the established objectives. The objective is to facilitate the application of the activities and tasks described in Clause 10 (Architecture Elaboration) of the 42020 standard, thereby enhancing comprehension and implementation.

The following specific objectives have been established:

- Map the activities and tasks of Clause 10 (Architecture Elaboration) of the 42020 standard, addressing key concepts that guide this process, such as architectural strategies and approaches, architectural views, architectural viewpoints, types, and styles of architecture, architectural models, and architectural requirements.
- Create a process that aims to reduce difficulties in understanding concepts, properties, and principles related to software architecture elaboration.
- Develop a software tool that allows stakeholders involved in the development and use of software architectures to utilize the developed process for elaborating and managing the architecture, as well as assisting in its validation.
- Publicly release the developed process and computer program, enabling users to access, modify, and contribute collaboratively to the evolution of the created artifacts, promoting transparency, participation, and active community contribution.

1.4 Methodology

An exploratory study will investigate Clause 10 (Architecture Elaboration process) of the 42020 standard, focusing on its seven activities and respective tasks to develop a streamlined architecture elaboration process. The research aims to deepen understanding, formulate hypotheses, or explicitly articulate findings (GIL, 2002). Therefore, comprehensive investigations will explore connections between Clause 10 and established software architectural practices, challenges in software development, and organizational case studies.

A literature review on software architecture will encompass standards and norms related to the Architecture Elaboration process. Search parameters will be limited to sources published within two years preceding the standard's release, available in English via digital databases including IEEE Xplore, ACM, Scopus, and ScienceDirect. Covering the period from 2017 to 2023 offers a contemporary perspective on advancements in modeling techniques, tools, and methodologies, while acknowledging earlier seminal works crucial to understanding the historical context of software architecture modeling and communication.

Given the artifact's creation (an adapted process incorporating Clause 10), particularly its practical application, the Design Science Research (DSR) approach will guide this study. According to Hevner et al. (2004), DSR is a research methodology aimed at generating and validating new knowledge in information systems through the design and implementation of artifacts addressing practical problems. The objective is to develop and assess innovative solutions applicable across various contexts.

When creating a process (an innovative artifact), there is a consequent advancement in the state of the art. In this specific case, by designing both a process and a tool that supports it, and by publishing these results, there is sufficient evidence that something innovative has been developed. This innovation, in itself, validates the advancement of the state of the art, which further justifies the choice of DSR as the research method.

Figure 1 illustrates a graphical representation outlining the application of these guidelines. Utilizing the 42020 standard as a knowledge base (rigor), a tailored architecture elaboration process (artifact) aims to mitigate challenges in comprehending architectural concepts, properties, and principles (relevance). Hevner et al. (2004) propose seven guidelines for a complete and effective DSR. Summarizing, this research needs to involve the creation of an innovative and purposeful artifact (1) to solve a specific problem (2). The artifact must be useful and comprehensively evaluated (3). Its novelty is essential as it should address an unresolved problem or do so more effectively (4). The artifact must be rigorously defined, formally represented, and coherent (5). The creation process incorporates a research process to find an effective solution (6). The results of the DSR should be effectively communicated to academic, technical, and managerial audiences (7).

Several methods can be employed to evaluate artifacts; however, this research has opted for a Proof of Concept. Although less rigorous from a methodological standpoint, the Proof of Concept allows for an initial assessment of the artifact, facilitating the identification of potential improvements and necessary adjustments before application in broader contexts. The research, illustrated in Figure 1, follows an iterative and incremental process, enabling the use of this preliminary approach to raise relevant unanswered questions and generate new inquiries and problems. Additionally, the study aims to conduct case studies and experiments as complementary evaluations of the artifact, utilizing criteria such as usability, comprehensibility, and ease of learning for its enhancement.

Upon initial artifact development, an application will be designed to support its application in a proof of concept, aiding in software architecture documentation, validation, and review. While this research does not focus on architecture evaluation, methods and techniques to evaluate software architecture documentation for compliance with 42020 and ISO/IEC 25010 standards, suitability for intended use, and support for architectural analysis will be established (CLEMENTS et al., 2010).

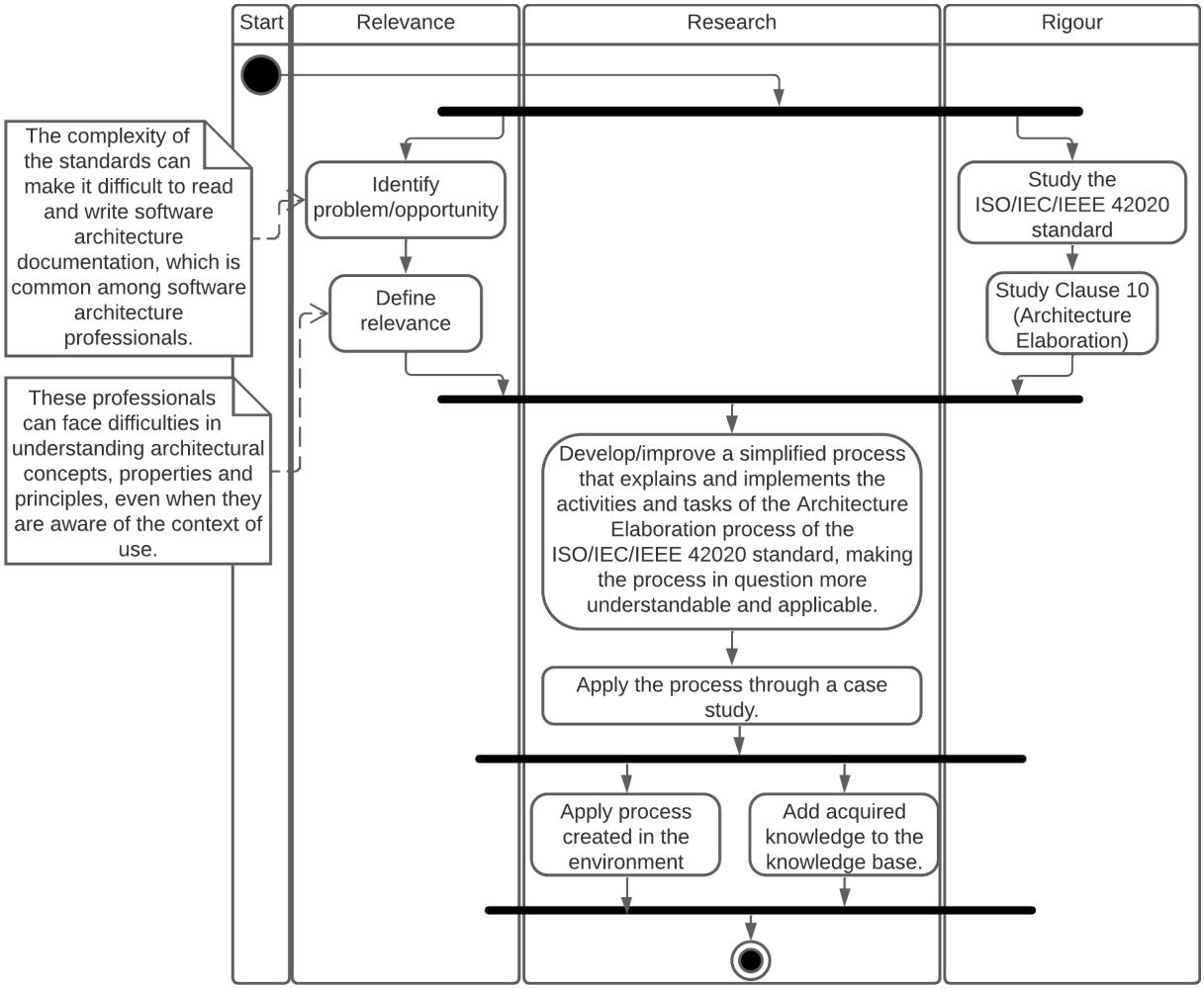


Figure 1 – Graphical Scheme of the Research

1.5 Structure of the work

Following the introduction, Chapter 2, "A Review on the Capacities of the 42020:2019 Standard for Architecture Elaboration of Software and Systems," provides a comprehensive overview of the ISO/IEC/IEEE 42020 standard. It details the specific processes described within the standard and examines its interactions with other related standards, establishing a critical review of its capabilities and limitations.

Chapter 3, "A Conceptual Review of the Relationship between ISO/IEC/IEEE 42010 and ISO/IEC/IEEE 42020," delves deeper into the theoretical aspects, exploring the evolution of standards and how they articulate the processes of architectural description. The present section emphasizes the linkage and distinctions between these standards to better understand their integration and application in architectural practices.

Chapter 4, "A Process for the Clause Architecture Elaboration of ISO/IEC/IEEE 42020," introduces a tailored process designed specifically for the Clause Architecture Elaboration. It systematically breaks down each step of the process, from preparation to supervision, providing a granular view of the tasks and activities involved. As such, this chapter is pivotal as it translates

theoretical insights into actionable procedures.

Chapter 5, “ArchCE: A Software to Generate Architecture Documentation Based on ISO/IEC/IEEE 42020,” shifts focus to a practical application by introducing and detailing ArchCE, a software tool developed to support the architecture elaboration process. It discusses the functionalities of the software and how it aligns with the processes outlined in the previous chapters.

The thesis concludes in the Chapter 6, “Conclusion”, which synthesizes the findings, discusses the implications, and suggests directions for future research. As such, this chapter aims to encapsulate the contributions of the thesis and its relevance to the field of software architecture.

Additionally, the document contains a section (6.4) on “Thesis-Related Publications”, which lists relevant publications and submissions related to the research, further illustrating the academic and practical contributions of the thesis. this comprehensive structure ensures a logical flow from theoretical exploration to practical application, encompassing a full spectrum of research, development, and evaluation within the domain of software architecture elaboration.

2

A Review on the Capacities of the ISO/IEC/IEEE 42020:2019 Standard for Architecture Elaboration of Software and Systems

The 42020 is an international standard that provides guidelines for the development of software architectures and systems. The 42020 describes a set of processes for governance and management of a software architecture or a collection of software architectures, as well as for architecting their entities (SANTOS; MISRA; SOARES, 2020). As such, this standard complements the architecture-related processes identified in 42010, 15288, 12207 and ISO 15704 (henceforth, 15704) (IEEE, 2019).

The capabilities of the software and system architecture elaboration will be explored in this chapter. To provide a clear roadmap, the chapter is structured as follows. First, an overview section explores the standard's purpose and key features, offering a glimpse into the six architecture processes it outlines. Following this, a section present these processes, categorizing them strategically (governance, management), primarily (conceptualization, elaboration, evaluation), and in terms of support (enablement). Each process will be briefly explained, outlining its purpose, activities, tasks, and resulting outcomes. To illustrate the standard's position within the broader landscape, a relationship section explores how 42020 interacts with other relevant standards. As such, this section will elucidate how these standards work together to facilitate software and system architecture development. Finally, the chapter concludes by summarizing the key takeaways and emphasizing the significance of the 42020 standard.

2.1 Overview of 42020

The 42020 standard, Architecture Processes, outlines six processes related to architecture: Architecture Governance, Architecture Management, Architecture Conceptualization, Architecture Evaluation, Architecture Elaboration, and Architecture Enablement, as shown in Table 1.

Process	Purpose
Architecture Governance	Aligning architectures with company objectives, policies and strategies.
Architecture Management	Implement the *Architecture Governance* process.
Architecture Conceptualization	Identify solutions that address the concerns of stakeholders.
Architecture Evaluation	Determine whether the proposed architecture sufficiently addresses the concerns of stakeholders.
Architecture Elaboration	Describing and creating documentation as well as applying modeling.
Architecture Enablement	Providing support to the other architecture processes so that they function correctly.

Table 1 – Process and purpose

Two related processes are fundamental for information exchange between the 12207 and 15288 standards: Architecture Management and Architecture Governance. As shown in Figure 2, the Architecture Management process is directly related to the entire software lifecycle established by these standards. It handles and coordinates interactions arising from non-technical processes. The Architecture Governance process provides guidelines for the development of architecture models that can be applied to various systems or an architecture portfolio (MARTIN, 2018).

The 42020 standard states that its objective is to establish a pattern of execution for software and systems architecture processes that enables its utilization in a variety of contexts and situations (IEEE, 2019). In a broader context, it facilitates negotiation in systems, products, and services.

2.2 ISO/IEC/IEEE 42020 processes descriptions

The processes will be explored in greater depth in this section, detailing their purposes, expected outcomes, activities, tasks, implementation, and generated documents. The objective is

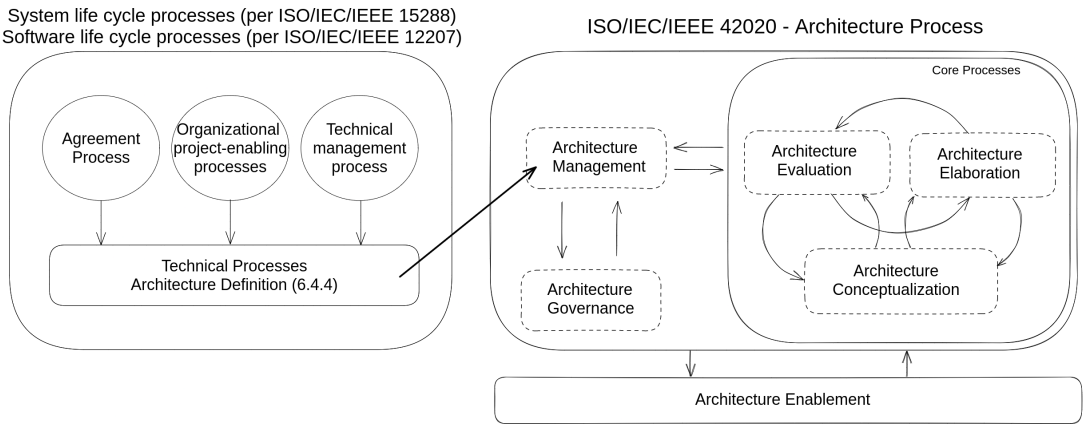


Figure 2 – Relationship between 42020 standard with ISO/IEC/EEE 12207 and 15288 standards

to develop a manual that explains the specific aim of each process while also demonstrating their interactions. Thus, it is expected to foster greater engagement with the processes among software and systems developers and architects.

Divided into three groups (strategic processes, primary processes, and support processes), the objective is to simplify communication between them by decomposing business decisions, architecture development, and support activities, as shown in Figure 3.

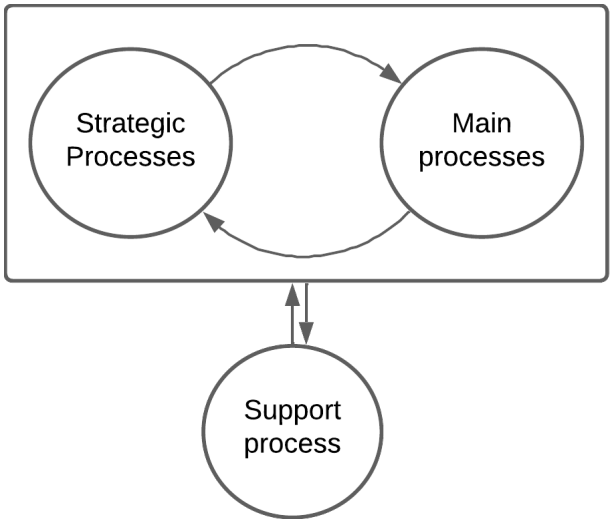


Figure 3 – Macro vision of the processes

2.2.1 Strategic processes

The processes Architecture governance and Architecture Management are classified as strategic processes in this master's thesis due to the nature of their activities. These processes are responsible for making decisions and guiding the Primary Processes in architecture creation. They may involve financial, political, technical, or organizational decisions from stakeholders, identified and documented during these processes' activities. Thus, all presented data are used

to create governance guidelines and directions to be implemented during management (IEEE, 2019).

Architecture Governance establishes governance guidelines and directions, while Architecture Management defines the management plans and instructions used to conduct the Primary Processes. It is important to note that stakeholders actively participate in the creation of the architecture, which, in turn, affects how the architecture will be managed. Strategic processes focus on decision-making and involve tasks requiring a high level of intellectual capacity, impacting an architecture or a portfolio of architectures.

Architecture Governance

The primary objective of this process is to establish and maintain reusable architecture governance models and frameworks. These frameworks will be responsible for maintaining a collection of software or system architectures. These collections may consist of various interrelated architectures or a single architecture aligned with organizational strategies, objectives, and policies (IEEE, 2019). The impact will also be significant on how the organization is structured, making decisions not only for technological elements but also for behavioral elements within the organization. Consequently, the process can establish or modify the organization's functioning and communication.

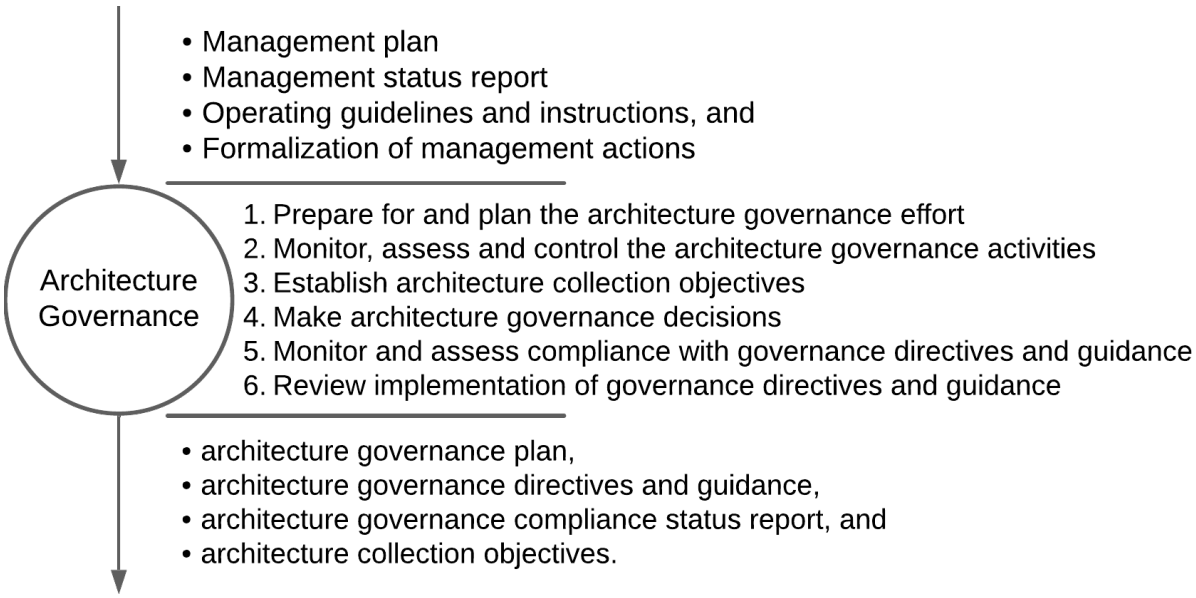


Figure 4 – Architecture governance process sketch

As illustrated in Figure 4, the Architecture Governance process is comprised of four inputs (work products), four outputs (work products), and six activities. Additionally, there are a total of forty-nine tasks, each of which is guided by principles. An organizational authority must ensure that the activities are carried out according to the established principles. The authority, sometimes referred to as the "Design Authority", is responsible for governing according to architectural

principles and escalating issues when necessary. The results of architecture governance are applicable throughout the organization, while the results of architecture management are applicable across the entire collection of architectures.

Architecture Management

The purpose of Architecture Management is to implement architecture governance guidelines to achieve the objectives of the architecture collection in a timely, efficient, and effective manner (IEEE, 2019).

The activities in this process adhere to architecture governance guidelines and standards, maximizing value for the stakeholders identified as relevant, relative to the available resources. Architecture Management establishes management plans and necessary guidelines to conduct the Primary Processes. It monitors compliance with the plans and management instructions using execution plans and the status of the primary processes. The management instructions for architecture serve as a means to assist the Primary Processes in implementing governance guidelines and aligning with management needs for process execution (IEEE, 2019).

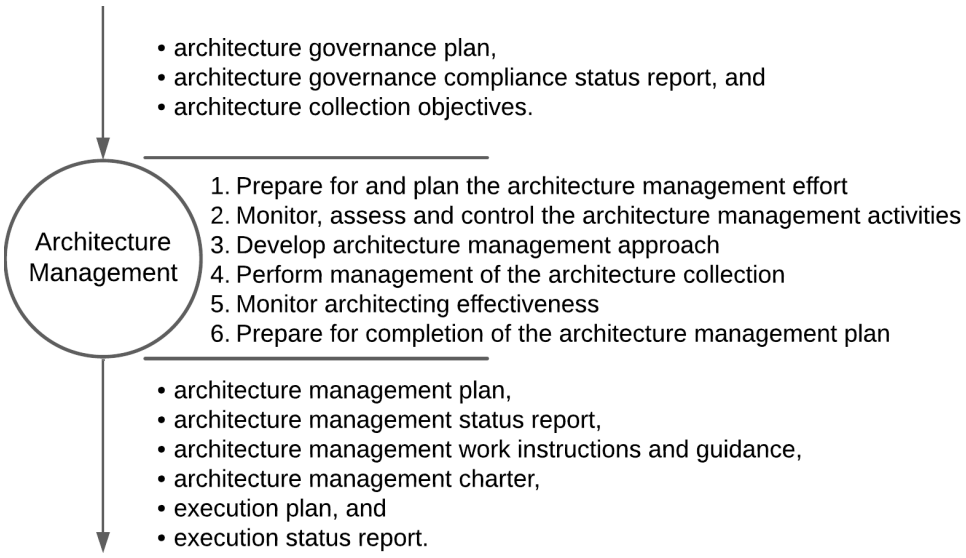


Figure 5 – Architecture management process sketch

As illustrated in Figure 5, the Architecture Management process is comprised of tree inputs (work products), six outputs (work products), and six activities. Additionally, there are a total of fifty-three tasks, each of which is guided by principles. The main characteristic is how this process manages decisions originating from the Architecture Governance process, the outcomes of the Primary Processes, and external factors influenced by the processes of the 15288 and 12207 standards. All outcomes (documents, reports, analyses, plans, historical information, defects, successes, among others) are stored in a repository for future reference, success cases, audits, and primarily for reuse in architectures.

2.2.2 Main processes

The processes categorized herein assist architects in characterizing the "problem space", elaborating architectural descriptions, defining architecture concepts, objectives, approaches, and, primarily, finding a solution for all the concerns raised. Elaboration, conceptualization, and evaluation of architectures provide descriptions of architectural solutions from a set of varied and detailed information derived from the Strategic Processes (IEEE, 2019). The architecture is elaborated and simultaneously refined in these three processes, through a cycle of requests and details, consistently validating concepts and alignment with objectives outlined by the Strategic Processes, thereby contributing to the architectural description. However, 42020 standard highlights the architecture elaboration process as it is here that most effort is concentrated in a description that aligns with the standard, being "sufficiently complete and correct".

Therefore, while in the Strategic Processes there is a focus on identifying concerns and issues, in the Primary Processes the aim is to find a response to stakeholders' concerns and architectural issues.

Architecture Conceptualization

The aim of this process is to identify the problem space and establish suitable solutions that address stakeholders' concerns, achieve architectural objectives, and meet relevant requirements.

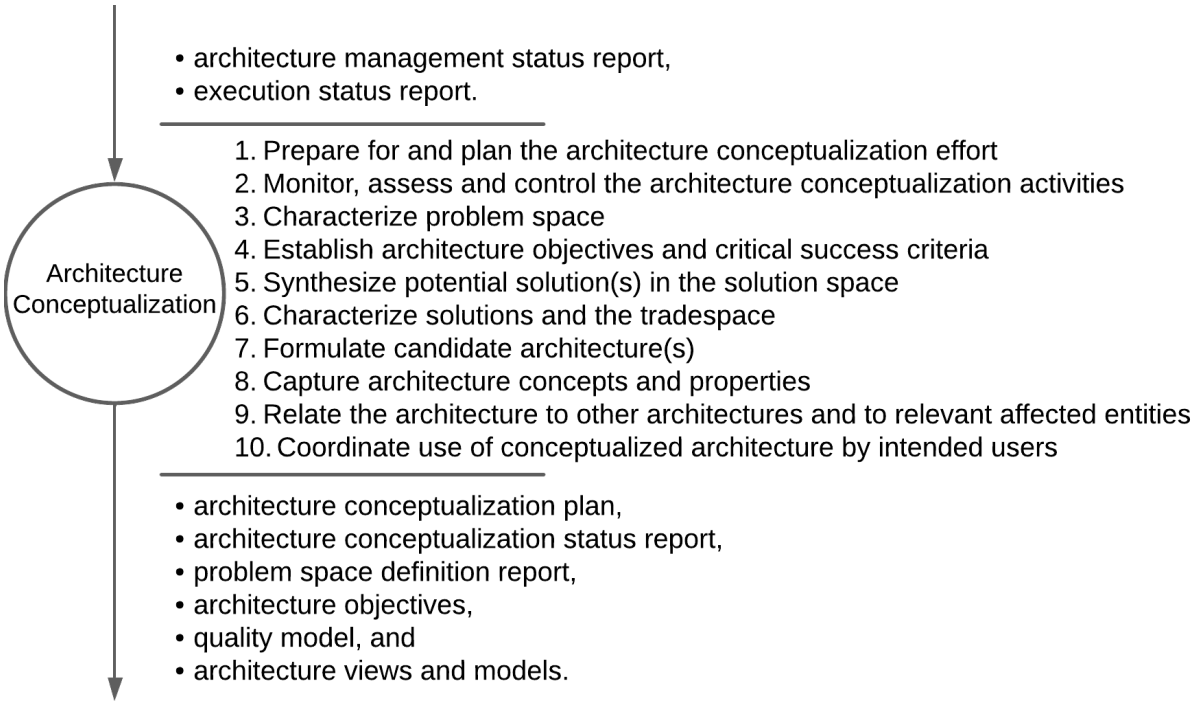


Figure 6 – Architecture conceptualization process sketch

As illustrated in Figure 6, the Architecture Conceptualization process is comprised of two inputs (work products), six outputs (work products), and ten activities. Additionally, there are a total of one hundred tasks, each of which is guided by principles.

The standard dictates that any process within the Primary Processes may be initiated by the Strategic Processes. Conceptualization presents the initial options of architectural descriptions and, with Evaluation, identifies the most feasible architectures to be passed on to Elaboration. The focus should be on solution identification, while also emphasizing a thorough understanding of the problem space. It also pertains to defining and establishing architecture objectives, as well as negotiating with key stakeholders on the prioritization of their concerns (IEEE, 2019; SANTOS; MISRA; SOARES, 2020).

Architecture Elaboration

Architecture Elaboration is activated multiple times throughout the architecture elaboration cycle to describe an architecture as completely as possible (IEEE, 2019). "Most complete" denotes a thorough and precise description of architectural viewpoints, perspectives, and models for intended purposes.

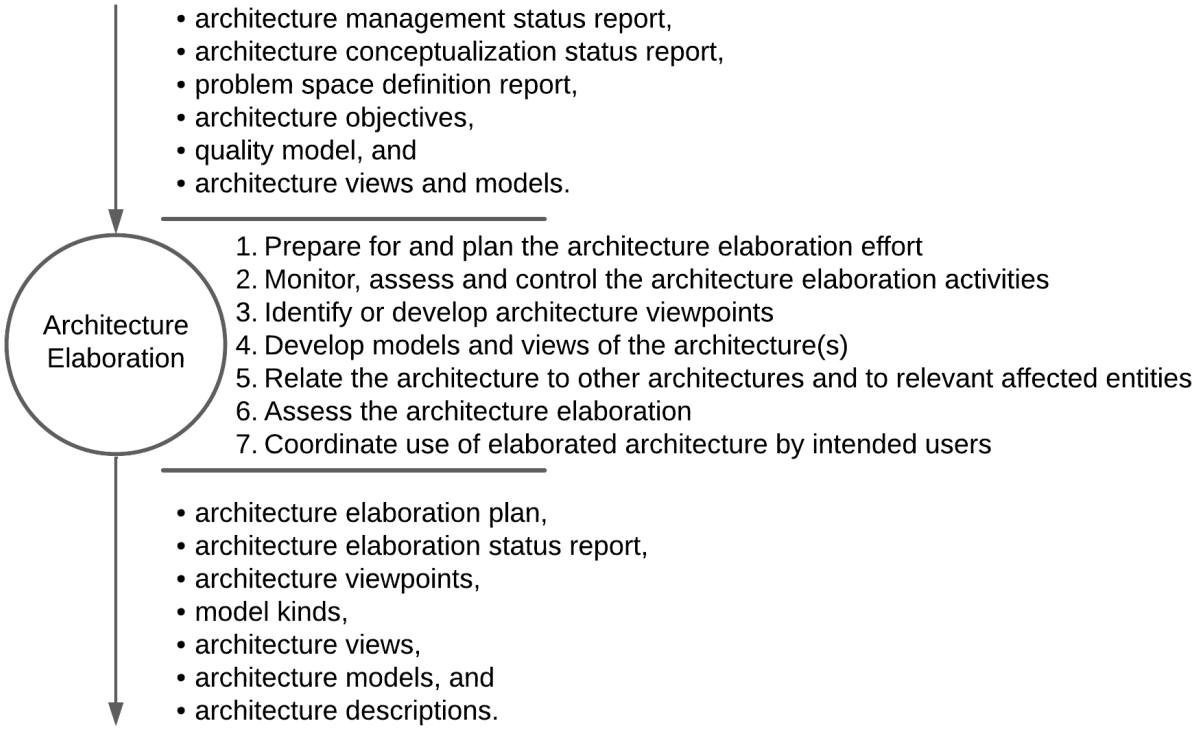


Figure 7 – Architecture elaboration process sketch

Although all three processes within the Primary Processes can describe an architecture, it is in this process that developers, architects, and software engineers truly work to create an architecture. Architecture frameworks available in the current software industry, such as the Unified Architecture Method (UAM) Framework, the Zachman Framework, The Open Group Architecture Framework, the DOD Architecture Framework, among others, are helpful during the elaboration of this process (MARTIN, 2021a). The rigor applied during architecture description establishes alignment with requirements, concerns, constraints, and the outline developed in the Architecture Conceptualization process.

As depicted in Figure 7, the Architecture Elaboration process is comprised of six inputs (work products), seven outputs (work products), and seven activities. Additionally, there are a total of seventy tasks, each of which is guided by principles. Notably, there is a direct correlation with the 42010 standard for implementation using concepts such as stakeholders, concerns, viewpoints, views, models, and architectural description, crucial for executing this process and achieving a comprehensive and well-developed architecture. Another relevant characteristic is the process having to deal with system architectures (described in 15288) and software architectures (described in 12207).

Architecture Evaluation

The process determines how well one or more architectures have achieved: their objectives, stakeholders' concerns, and relevant requirements.

The conclusion of its application reveals, among the considered architectures, which one is most relevant in addressing stakeholders' concerns, constraints, costs, risks, opportunities, tradeoffs, among others.

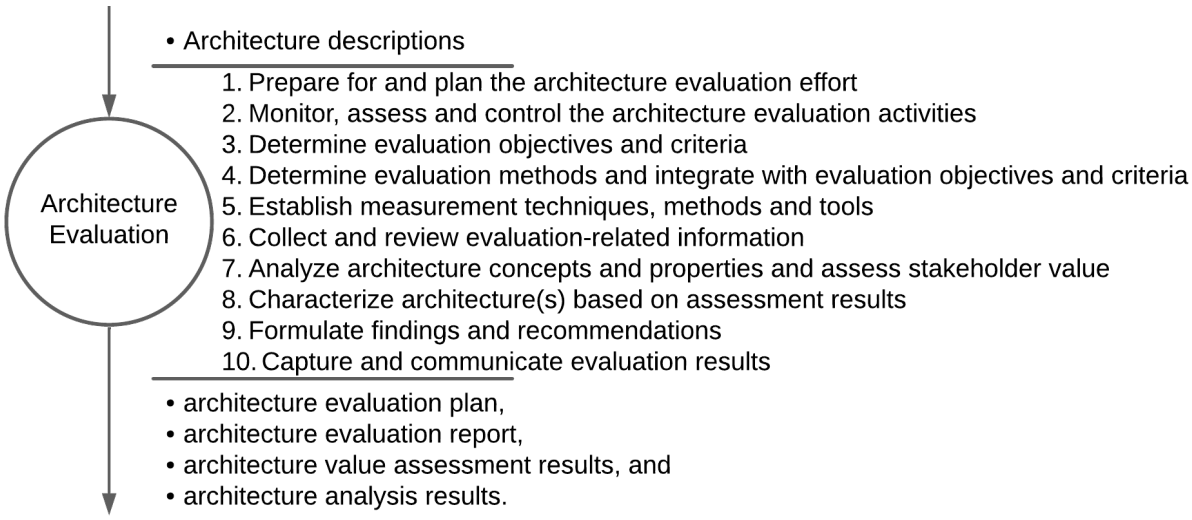


Figure 8 – Architecture evaluation process sketch

As depicted in Figure 8, the Architecture Elaboration process is comprised of one input (work product), four outputs (work products), and ten activities. Additionally, there are a total of ninety-two tasks, each of which is guided by principles. The direct connection with ISO/IEC/IEEE 42030 is central here. The process activities align with the concepts and principles of ISO/IEC/IEEE 42030, a generic reference framework for Architecture Evaluation, comprising evaluation objectives, value evaluation objectives, architectural analysis objectives and factors, evaluation, and analysis.

2.2.3 Support process

Architecture Enablement

Architecture Enablement develops, maintains, and enhances the skills, services, and resources necessary for the execution of other processes within this standard.

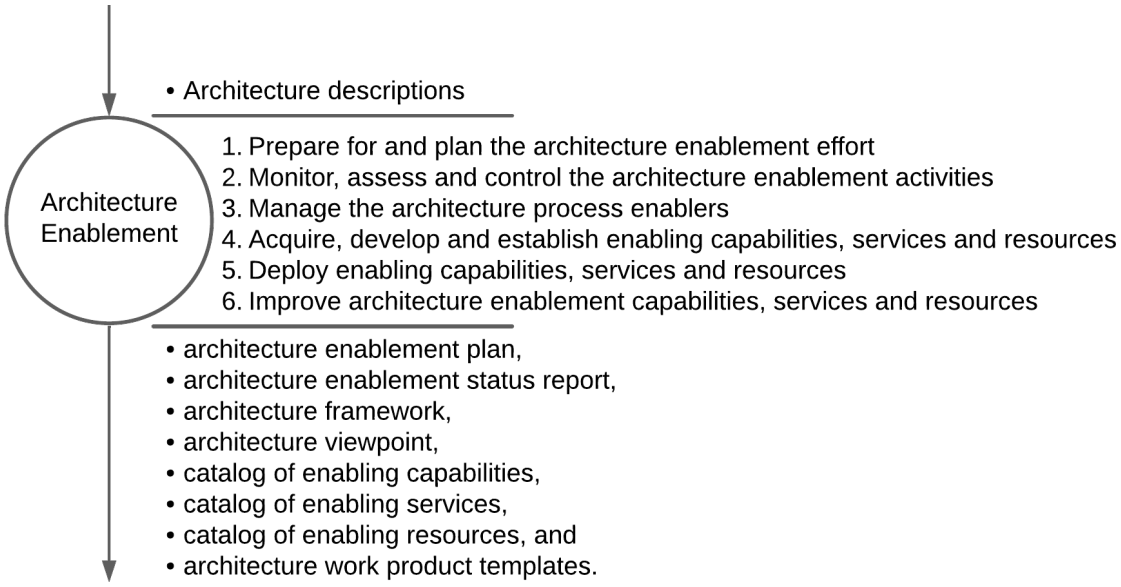


Figure 9 – Architecture enablement process sketch

As depicted in Figure 9, the Architecture Enablement process is comprised of one input (work product), eight outputs (work products), and six activities. Additionally, there are a total of fifty-two tasks, each of which is guided by principles.

The activity “Manage the architecture process enablers”, consisting of six tasks, is relevant for managing decisions, risks, changes, repositories/libraries, quality, and infrastructure, with processes such as “Decision, risk, configuration, information, and quality management” from 15288 and 12207 serving as enablers for this process.

2.3 Relationship with other standards

According to 42020, there is a relationship between 15288 and 12207 in four cases:

- Processes from 15288 or 12207 provide information for processes like Architecture Conceptualization and Evaluation in 42020, for instance, to identify necessary interfaces for validation and compliance with architecture-related constraints.
- Processes from 42020, such as Architecture Elaboration, provide information to processes from 15288 or 12207, such as the description of a validated architecture.
- A process from 42020 is implemented within a specific process from 15288 and/or 12207.

- A process from 15288 and/or 12207 is performed within a process from 42020.

There are also several points regarding the relationship between 42010, 42020, and ISO/IEC/IEEE 42030:

- 42020 expands concepts beyond how they are explained in their original standards. For example, concepts of "Architecture Framework" had to be expanded in 42020 since 42010 presents the concept but not the relationship between standards and existing frameworks.
- 42010 is directly related to the activities of the Architecture Elaboration process in 42020. While the former specifies requirements in architecture description elements, the latter specifies requirements in architecture elaboration activities.
- The Architecture Evaluation process defined in 42020 is directly related to the application guidance of ISO/IEC/IEEE 42030. The activity "Determine Evaluation Criteria and Objectives" (9.4.3) is supported by a generic framework that helps determine to what extent stakeholders' concerns are met using concepts of value assessment and architectural analysis.

2.4 Summary

The 42020 standard provides guidelines for the management and description of an architecture or set of architectures in accordance with the 42010 standard. Collectively, they facilitate the implementation of the process that defines software or system architecture(s) as outlined in the 15288 and 12207 standards.

The objective of this chapter is not to provide a comprehensive analysis of the 42020 standard. Instead, it offers a general overview of the concepts, principles, and theories related to the use of the standard and the processes covered in the standard. The provided overview is designed to assist a diverse range of professionals and organizations in their use or study of this standard. It does so by elucidating its key elements in a clear and concise manner.

The following chapter will examine the necessity for frequent reevaluation of the 42020 and 42010 standards, which are often perceived as challenging to interpret. The necessity for reevaluation is driven by the emergence of new insights and interpretations, which in turn reveal the underlying factors that prompted the development of 42020. The primary focus of this study is on the 42020 standard and its Architecture Elaboration process, which is responsible for the actual development and management of Architecture Descriptions.

3

A Conceptual Review of the Relationship between ISO/IEC/IEEE 42010 and ISO/IEC/IEEE 42020

The 42020 applies the concepts and guidelines of 42010, which specifies the requirements for the structure and expression of an Architectural Description (AD) for various entities, including software, systems, companies, systems of systems and families of systems (IEEE, 2019; IEEE, 2022). While 42010 addresses what an Architectural Description should contain and how it should be expressed, 42020 presents how to develop and manage an Architectural Description. Both standards are important for documenting software architectures and systems based on scientifically referenced standards and rules (SIEVI-KORTE; BEECHAM; RICHARDSON, 2019).

By synthesizing information from various authors, including Martin (2018), Júnior, Misra and Soares (2019), Junior, Misra and Soares (2019), Wohlrab et al. (2019), Martin (2021b), and considering the IEEE (2022) and IEEE (2019), several dimensions of strength and weakness can be observed in their interaction. The lack of specific guidelines for application in organizational or project contexts in both standards implies a need for autonomous judgment on the part of users or the search for additional sources of effective guidance. Conversely, both standards are notable for establishing a common terminology and a solid conceptual framework for software architecture. The attribute of establishing a common terminology and a solid conceptual framework for software architecture allows users to employ these standards together, facilitating communication and collaboration with other professionals by supporting a uniform language and understanding. However, both standards face the challenge of not prescribing specific architectural methods, techniques, or tools, which requires users to select or develop customized approaches according to their needs and preferences. Finally, a strength shared by both standards lies in the specification of requirements that promote quality and consistency in architectural processes and products. These requirements enable users to use them together to evaluate and improve their architectures

based on objective criteria and metrics.

Standards 42010 and 42020 play an important role in the software application development cycle, including the very structuring and ordering of the organization in question. They describe the structure, behaviour, and properties of an entity and its components, as well as the relationships between them (GARLAN; SHAW, 1993). Standards also provide a basis for analyzing, designing, implementing, verifying and evolving an architecture (MARTIN, 2015). To express and communicate software architecture in a standardized way, 42010 defines a standard for the structure and expression of an Architectural Description for various entities. Standards 42020 provides guidelines for developing software architectures using Architectural Description. However, 42010 does not address how an Architectural Description should be developed and managed throughout the lifecycle of an entity, which is the case of 42020.

From here, it will be analyzed which factors present (or absent) in 42010 led to the need for the 42020 proposal, especially to understand in which 42020 process the development and management of an Architectural Description takes place. In this sense, as well as discussing the relationships between these standards, this chapter will promote a study of Architectural Descriptions in the context of the development of a Software Architecture presented by 42020.

For the purposes of systematizing and understanding, the following sections are outlined to achieve the chapter objective: (3.1) the contexts of both standards are presented, discussing related works that deal with their applications and associated practices; (3.2) a practical approach is proposed for describing Software Architectures, following the principles and guidelines of standard 42020; (3.3) the limits of each standard are identified and elucidated, clearly establishing the start and end points of their scopes in the context of Software Architectures; and (3.4) a critical analysis is proposed for the results obtained, accompanied by reflections on the applicability of the standards and practices, culminating in relevant conclusions for the advancement and improvement of practices related to Software Architecture.

3.1 Evolution of standards

Several studies emphasize the indispensability of providing support for describing and evolving software architectures, a particularly notable demand in the context of agile methodologies, as evidenced by various studies over the years Garlan and Shaw (1993), Maier, Emery and Hilliard (2001), Abrahamsson, Babar and Kruchten (2010), Santos, Misra and Soares (2020), Rocha, Misra and Soares (2023). The elaboration and utilization of software architecture have also progressed, as observed in Aldenhoven and Engelschall (2023) and Galster and Weyns (2023), approaching functional definitions similar to the original concept in Civil Engineering. In the field of Systems and Software Engineering, architecture is applied in different stages of system or software design, encompassing the set of instructions in a central processing unit, software modules at both high and low levels, extensive systems with interconnected services, the

overall structure of information technology systems in organizations, and may even encompass a product line and/or an acquisition object.

The confusion between architectural design and software or systems design has been a topic of study in the past decades (SOARES, 2010; ROCHA; MISRA; SOARES, 2023). The need for distinction and clarity has arisen over time. This viewpoint has been integrated into standards such as 12207, 15288, and 15704, and subsequently expanded in standards 42010 and 42020. These standards emphasize that software architecture not only guides software design and specification but also finds application in various areas, including enterprise portfolio management, operational support, business development, mission planning, capacity management, and process assessment. Its influence is not limited to design decisions but extends to those of applied areas (MARTIN, 2015; BAABAD et al., 2020).

The "software architecture design" concerns the conception and definition of the fundamental structure of a software system, encompassing decision-making across various aspects of the system or software lifecycle. This involves the interaction and organization of components, selection of architectural patterns and styles, and considering requirements, concerns, and constraints of stakeholders to generate a solution that addresses the presented problem or opportunity (IEEE, 2015).

Although this perspective has been addressed through recent standards, initially the 15288 in 2015 and subsequently the 12207 in 2017, the distinction between architecture and design has been explored by various authors, as mentioned. The discussion on "architecture definition", the influence of civil architecture, the role of Architectural Descriptions, as well as the intrinsic relationship between software architecture design and software architecture descriptions has also been studied (MAIER; EMERY; HILLIARD, 2001). Both concepts are interconnected and play important roles in the development of complex software systems. IEEE Std 1471 was replaced in 2011 by 42010. Figure 10 describes the standards over the years and highlights the evolution through their titles.

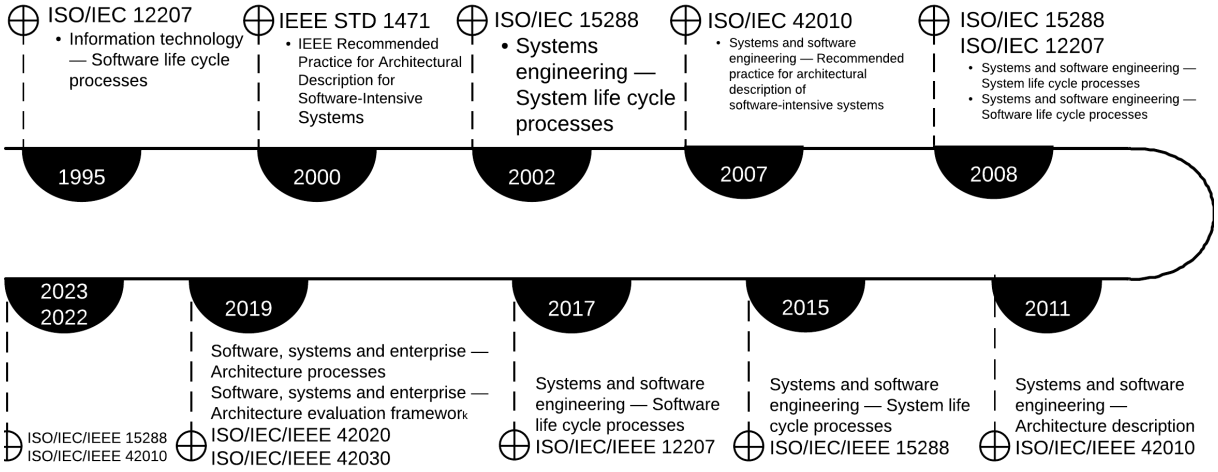


Figure 10 – Evolution of standards over the years

The discussion about architecture and design has also led to new approaches for describing and elaborating software architectures aligned with modern practices. After the update of 15288, which separated design and architecture in the context of systems and software engineering, there emerged a need for more comprehensive practices to elaborate software architectures. Thus, standards 42020 and 42030 were introduced (MARTIN, 2018).

These standards were developed to address the need for standardized terminology, methodology, and tools for designing and managing Software Architectures. Their relevance lies in providing a conceptual framework for Software Architectures and Architecture Descriptions for various entities, including software, systems, organizations, systems of systems, and system families (IEEE, 2019; IEEE, 2022). They assist organizations in developing architectures that satisfy stakeholders and align with organizational objectives. Additionally, they aid in managing architectures throughout the software lifecycle.

To explain the connection and cooperation between these standards, it is possible to use the UML package diagram, as illustrated in Figure 11. For a complete understanding of this diagram, it is necessary to define the following stereotypes presented in Table 2.

Stereotypes	Definition
«norm»	It indicates that a package represents a specific norm or standard.
«subdiscipline»	It indicates that a package represents a subdiscipline or a subdivision of a broader study field.
«discipline»	It indicates that a package represents a discipline or a broader study field.
«refinement»	It indicates a refinement relationship between packages, where the dependent package details or enhances the independent package.
«extends»	It indicates an extension relationship between packages, where the dependent package extends or complements the independent package.
«conformance»	It indicates a conformance relationship between packages, where the dependent package follows or adheres to the independent package.

Table 2 – Stereotypes definition

Each package represents a standard or a set of standards. The *«norm»* package 12207 represents the standard defining software lifecycle processes. The *«discipline»* package "Systems and Software Engineering" represents the set of standards addressing common aspects of systems and software engineering, such as the 15288, which defines system lifecycle processes. The *«subdiscipline»* Architecture package represents the subset of standard 42010 that defines system or software architecture concepts and terminology.

The dashed arrows indicate dependencies between the packages. The *«refines»* arrow indicates that the dependent package refines or elaborates on the independent package. Thus, the 12207 package refines the Systems and Software Engineering package by specifying software

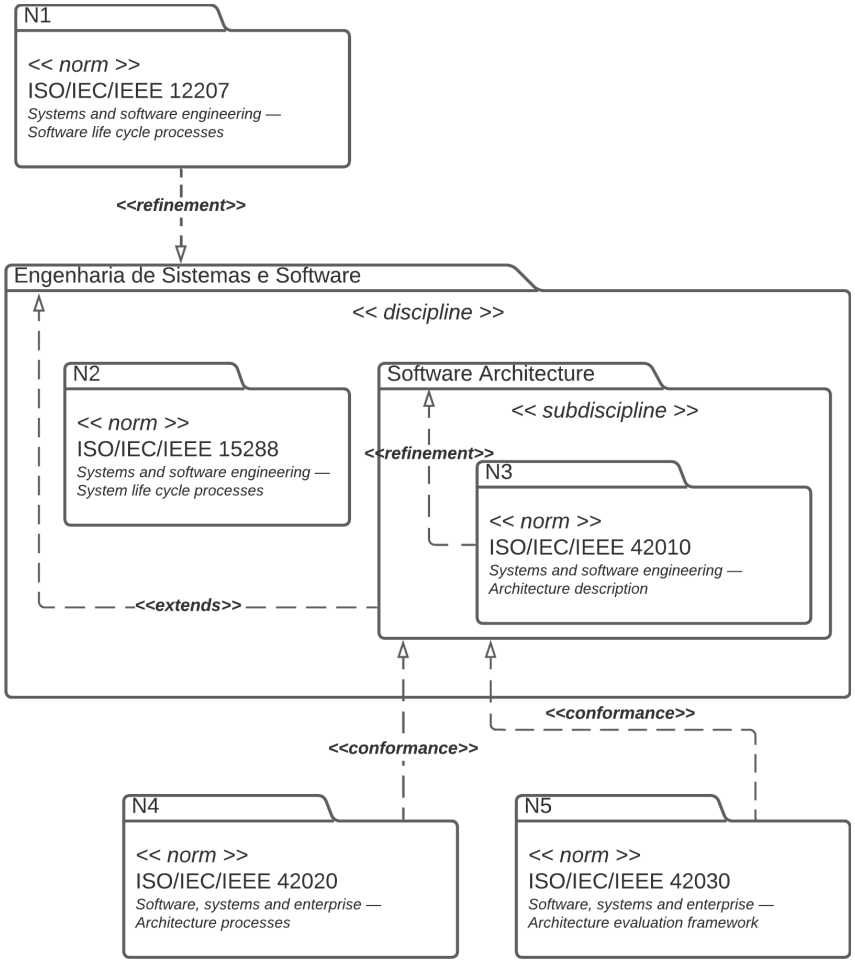


Figure 11 – Connection and cooperation between standards

lifecycle processes. The *«extends»* arrow indicates that the dependent package extends or supplements the independent package. Thus, the Architecture package extends the Systems and Software Engineering package by defining system or software architecture concepts and terminology. The *«conforms»* arrow indicates that the dependent package conforms to or follows the independent package. The 42020 and ISO/IEC/IEEE 42030 packages conform to the Architecture package by defining system or software architecture principles.

It is worth noting that these standards provide guidelines that can be applied to various types of software, regardless of size. While they do not constitute a definitive solution, they represent a repository of empirical studies and best practices.

In the subsequent sections, the points of convergence and divergence between the standards are examined, as well as the potential challenges in implementing these guidelines in diverse environments.

3.1.1 Every Architectural Description expresses an architecture

Architecture – a concept – what the act of architecting produces – a set of documents – and how to describe these documents – a standard – are not the same thing. Since IEEE Std

1471, efforts have focused on standards for describing architectures, not the actual architecture of the system or software (MAIER; EMERY; HILLIARD, 2001). It is in this manner that 42010, driven by modern Model-Based Systems Engineering (MBSE) practices, focuses on how to describe architectures based on stakeholder concerns, viewpoints, views, and models, not on standardizing processes to create them (MARTIN, 2021b). The process for creating architectures, namely producing a set of documents capable of expressing a concept, is seen in the Architecture Elaboration process of 42020, which establishes a foundation for sound architectural practices and terminology. Analogous to civil construction, the idea is familiar regarding communities having architectural standards that restrict how buildings are constructed; at the same time, they have standards for how building blueprints are described, meaning there is a clear distinction between blueprint and construction conventions (MAIER; EMERY; HILLIARD, 2001).

Considering the standardization proposed by 42010, an Architectural Description is composed of Architecture Description elements, or as the standard itself summarizes: Architectural Description element (IEEE, 2022). Each Architectural Description element is a representation or instance of various architectural concepts within an Architectural Description document, which typically includes several elements, each of which incorporates one or more of these architectural concepts. These architectural concepts are outlined in the standard; however, they are better defined in Martin (2021b):

- Architectural Description identification, overview and additional information
- Identification of the stakeholders of the entity of interest
- Identification of relevant concerns
- Identification of the relevant aspects of the entity
- Identification of relevant stakeholder perspectives
- A definition or reference for each architecture viewpoint used in AD
- Architecture view(s) for each architecture viewpoint used
- Architecture View Components that are shared between architecture views
- A record of known inconsistencies between the contents required of AD
- Applicable Architectural Description matches and Architectural Description matching methods
- Applicable Architectural Description element matches and Architectural Description element matching methods
- Justifications for the architectural decisions made

Standard 42010 is more focused on standardizing the structure and content of an Architectural Description document than on standardizing the process or methods for describing or documenting the architecture itself. Such inferences can be drawn in a number of ways, according to (MARTIN, 2021b; IEEE, 2022; JUNIOR; MISRA; SOARES, 2019):

1. **Emphasis on Document Structures:** The standard defines detailed requirements for the content of an architecture description document. For instance, it mandates the inclusion of elements such as stakeholder identification, architectural concerns, architectural views, and justifications for architectural decisions. Such an approach indicates that standardization is focused on the document structure.

2. **Non-Standardization of Description Process:** The standard does not standardize the process of elaborating an architecture description or documentation. It does not define how people should gather information, make decisions, or perform specific activities to create the architecture description. Instead, the standard concentrates on establishing what should be present in the outcome (the document).

3. **Emphasis on Document Aspects:** The standard also underscores the importance of aspects such as the organization of models and views within the document, the definition of architectural viewpoints, and the creation of architectural frameworks in a grid or matrix. All of this pertains to how information should be presented in a document, not how the architecture should be developed or documented per se.

Therefore, the focus of 42010 is to provide detailed guidelines for creating a comprehensive and well-structured architecture description document, leaving the choice of specific methods and processes to collect, analyze, and document system and software architecture according to the guidelines of 42020.

3.1.2 Every architecture is expressed by an Architecture Description

Analogous to civil construction, it is necessary to create blueprints with elements defined by 42010. The questions now arise: How to elaborate? How to manage this elaboration? How to govern the process of documenting an architecture? It is at this point that 42020, according to (MARTIN, 2018; IEEE, 2019), relates to the practices of elaborating systems or software architectures. These practices refer to the process of conceiving, defining, expressing, documenting, communicating, certifying, maintaining, and improving an architecture throughout the lifecycle of a system or software (IEEE, 2019). The 42020 expands the scope of 42010 and addresses the processes and practices involved in creating and managing systems or software architectures.

By combining information from (IEEE, 2019), (SANTOS; MISRA; SOARES, 2020), and (MARTIN, 2021a), a broader idea about 42020 can be formed as follows:

- Standard definition: The 42020 is an international standard that addresses processes related to architecture, specifically in the context of developing and using systems architectures.
- Scope: The standard covers a broad set of processes, which are divided into six main categories: governance, management, conceptualization, evaluation, elaboration, and enablement. Each of these categories represents an important aspect in the lifecycle of systems architecture.
- Purpose: The purpose of 42020 is to establish a set of guidelines and standards to guide the creation, management, and effective use of systems architectures. It aims to provide a framework to ensure that architectural practices are aligned with the organization's strategic objectives and meet stakeholders' needs.
- Interconnection with other standards: The 42020 is related to other standards, such as 12207, 15288, 15704, 42010, and 42030. 42020 standard integrates into the broader context of systems and software engineering by providing specific guidelines for architecture within that context.
- Expanding the Role of Architecture: 42020 standard seeks to expand the role of architecture in the systems or software engineering process, especially in areas such as governance and architecture management. Consequently, architecture is not merely a technical description, but rather assumes a strategic and managerial role within organizations.
- Impact on the occupation and education: The standard has a significant impact on the systems engineering profession, affecting products such as systems engineering manuals and certification exams. Additionally, it influences systems engineering education by providing a set of standardized practices to be followed.
- Role in standardization: The standard is part of the set of standards developed by ISO/IEC/IEEE to standardize and improve practices in systems engineering and architecture.

From here, this dissertation thesis turns to the Architecture Elaboration process of 42020, not because it is the most important, since the standard provides information on various architecture-related processes. The Architecture Elaboration process is one of the six processes mentioned in this standard, and each process has its purpose and specific role in the development and use of systems or software architectures (IEEE, 2019). The reason is the architecture elaboration phase, where architectural concepts are formalized using the elements established by 42010 (MARTIN, 2021a). The elaboration of architectural models represents a pivotal stage in the architectural development process. These models serve as a means of communication and implementation for systems or software architectures.

3.2 The Process of Describing an Architecture

The software architecture elaboration process must be robust, scalable, and well-documented, constituting a fundamental step in software development. It involves starting from a high-level architecture design and refining it into a detailed plan that guides software implementation. This process not only ensures that the architecture aligns with stakeholders' needs and requirements but also provides the necessary guidance for developers to construct the software effectively (WOHLRAB et al., 2019). To guide this detailed process, 42020 provides a structured framework for elaborating software or system architectures. This section will explore the Architecture Elaboration process by combining insights from both (IEEE, 2019) and (MARTIN, 2018), illustrating its practical relevance with real-world examples.

3.2.1 Architecture Elaboration

The primary objective of the Architecture Elaboration process is to describe or document a software architecture in a sufficiently complete and accurate manner for its intended uses (IEEE, 2019). In addition, this process entails capturing the fundamental concepts and properties of the architecture, aligning it with relevant requirements and design characteristics, and ensuring that it can guide the development and evolution of the entity being architected (MARTIN, 2018). In other words, it ensures that the architecture is well-defined, well-documented, and suitable for its intended purpose.

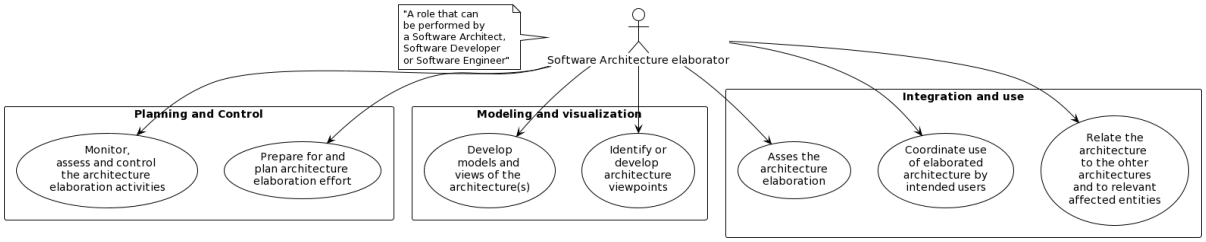


Figure 12 – Activities of the Architecture Elaboration

The specific objectives of this process include (IEEE, 2019): 1) developing appropriate architecture viewpoints and architecture metamodels for creating architecture views and suitable architecture models; 2) Capturing architecture views and architecture models with an appropriate level of detail for their specific purposes; 3) expressing architecture views and architecture models accurately and comprehensively to represent key aspects of the architecture; 4) ensuring that the architecture remains aligned with changes occurring during its development; 5) aligning the architecture with relevant requirements, design characteristics, and other related architectures; 6) providing a well-managed and controlled architecture description; 7) identifying and mitigating risks associated with architecture development.

This process involves seven activities, as depicted in Figure 12, each contributing to the refinement and completeness of the software architecture. As described in (IEEE, 2019), they are:

Prepare and plan architecture elaboration. Involves identifying potential users, issues to be addressed, stakeholder concerns, and intended architecture objectives. It also covers planning, resource allocation, and training.

Monitor, assess, and control architecture elaboration activities. This step involves continuous monitoring, reporting, and risk assessment throughout the elaboration process. It also emphasizes alignment with Architecture Governance and Architecture Management processes.

Identify or develop architecture viewpoints. Here, the process focuses on selecting, adapting, or developing viewpoints and model kinds based on stakeholder concerns and defining their purpose and scope.

Develop architecture models and views. This step involves defining architectural context, identifying entities and relationships to be modelled, selecting modelling methods and tools, and managing and modifying architecture descriptions based on the created models.

Relate the architecture to other architectures and relevant affected entities. Discusses how a new architecture relates to existing one(s) or documented elements, defines interfaces and interactions, and evaluates potential reuse and requirement allocation.

Assess architecture elaboration. Involves assessing architecture views and architectural models against intended objectives, validating against conceptualized architecture, ensuring adequacy, and addressing any discrepancies.

Coordinate the use of elaborated architecture with involved users. Focuses on preparing and delivering architecture documentation to users, monitoring its use, and incorporating feedback.

Concluding this concept, it can be said that the focus is to create a comprehensive and well-structured description of the architecture that serves as a valuable resource for all stakeholders involved in the development and evolution of the entity being architected.

3.2.2 Architectural Descriptions in the context of 42020

Based on (IEEE, 2019; MARTIN, 2018), Architecture Elaboration, which provides a structured approach to documenting architecture—and Architecture descriptions, which establish the foundation for creating consistent and well-structured architecture descriptions, are interrelated in the following ways:

- **Compliance with standards:** The Architecture Elaboration process, as per 42020, explicitly mentions that it must comply with 42010. This means that the principles and guidelines outlined in 42010 should be followed when carrying out architecture elaboration activities.
- **Guidance for elaboration:** The 42010 provides fundamental concepts and terminology essential for the effectiveness of architecture elaboration. It helps architects and professionals

understand how to structure Architectural Descriptions and select appropriate viewpoints and model types.

- **Alignment with stakeholder concerns:** 42010 and the 42020 Architecture Elaboration process emphasize the importance of addressing stakeholder concerns. The 42010 introduces the concept of architecture viewpoints, which are used to create specific architecture views that address stakeholders' concerns. The Architecture Elaboration process further extends this concept by detailing how to develop and refine these views.
- **Documentation and Alignment:** The 42010 lays the groundwork for documenting architecture descriptions, which includes considerations for viewpoints, models, views, aspects, and perspectives. The Architecture Elaboration process, in turn, focuses on the practical implementation of these documentation principles, ensuring that the architecture description remains aligned with constantly evolving requirements and objectives.

Standard 42010 provides general guidelines for the architecture description(s), while the Architecture Elaboration process offers a specific set of activities and tasks to refine and detail an architecture, generating a document in compliance with the standards and principles outlined.

3.3 What's left to start documenting?

The standards 42010 and 42020, which are involved with software or system architecture, are closely related but serve different purposes in the context of developing and documenting a software architecture (SANTOS; MISRA; SOARES, 2020). Table 3 presents eight general characteristics of the standards, where each characteristic represents a different purpose.

Characteristics	12207	15288	42010	42020	42030
1	●	❍	❍	●	●
2	❍	●	❍	●	●
3	❍	❍	●	❍	❍
4	❍	❍	❍	●	❍
5	❍	❍	❍	●	❍
6	❍	❍	●	❍	❍
7	❍	❍	❍	●	❍
8	❍	❍	❍	❍	●

Table 3 – Characteristic/Purpose of each standard

Table 3 shows characteristics defined as follows: (**1**) Software lifecycle processes; (**2**) System lifecycle processes; (**3**) Requirements for the structure and expression of an Architectural Description for various entities, including software, systems, enterprises, systems of systems, and

families of systems; (**4**) Guide the development of system and software architectures; (**5**) Establish a set of processes for governance and management of a collection of architectures and the architecture of entities; (**6**) Specify requirements that apply to the Architectural Description; (**7**) Evaluate architecture documentation; (**8**) Evaluate software and system architectures. Interesting conclusions can be drawn from Table 3:

- 42020 can handle system architectures, software architectures, and enterprise architectures as needed (IEEE, 2019). This suggests that 42020 was designed to align with the processes described in 12207 and 15288, ensuring that architecture activities are integrated into broader engineering processes for systems and software.
- 42010 and 42020 are integral components of a broader framework that includes 12207 and 15288. This framework ensures that activities related to architecture are well integrated into broader systems and software engineering processes, providing a structured approach to the development and management of complex systems and software architectures (IEEE, 2019).
- Regarding 42010, 42020, and 42030, the development of these standards has changed the view on software architecture practices, affirming the separation of concepts between architecture and design and emphasizing the concept that "every architecture is design, but not every design is architecture" (BOOCH, 2008; MARTIN, 2018; KUMAR; LOKKU; NATARAJAN, 2021b).

Complex but fundamental standards. One of the main strengths lies in the complementarity between the standards. Together, 42010 and 42020 provide a robust conceptual framework (42010) and a practical guide (42020) for creating and documenting system architectures (IEEE, 2022; IEEE, 2019). Both standards are designed to be flexible and adaptable to different contexts and types of architecture (MARTIN, 2021a). This allows organizations to choose relevant parts of each standard and adjust them according to their specific needs.

Another strong point is the improvement in communication and collaboration. By following the guidelines of 42010 and 42020, architecture teams can significantly enhance communication and collaboration in projects (HASSELBRING, 2018). Standardizing terminology and processes reduces ambiguity and facilitates understanding among stakeholders (IEEE, 2022).

However, software teams may initially find standards 42010 and 42020 complex due to the number of concepts and guidelines involved. This may discourage adoption, especially in small organizations or smaller-scale projects. Simplifying the language and providing more accessible examples could help overcome this obstacle (KOWALEWSKI, 2020).

Making the most of these standards requires an investment of time in understanding and implementing the guidelines. This may result in a steep learning curve, especially for teams inexperienced in system architecture. Proper training can help mitigate this challenge.

Lastly, the successful application of standards 42010 and 42020 requires effective coordination among different parts of the organization. This can be difficult to achieve in large and complex organizations, where multiple teams may be working on different parts of the architecture. Lack of coordination can lead to inconsistency and redundancy in architecture documentation.

Although standards 42010 and 42020 offer a solid framework and practical guidelines for system and software architecture, they require effort in learning and proper coordination to be fully effective. Understanding the strengths and weaknesses of these standards is crucial for making informed decisions about their adoption and implementation, leading to the need for further empirical studies to assess the real impact of adopting standards 42010 and 42020 in system and software architecture projects.

How to document? We have an approach, but it raises questions. Within the context of describing or documenting architecture, the Architecture Elaboration process, as described in (IEEE, 2019) and with studies from various authors (MARTIN, 2018; JUNIOR; MISRA; SOARES, 2019; MARTIN, 2021b; KUMAR; LOKKU; NATARAJAN, 2021b), raises several significant discussions.

First, there is a discussion regarding the need for a structured approach to architecture description. The Architecture Elaboration process emphasizes the importance of this, sparking debates on how such a structured approach can enhance architectural understanding and communication.

Another point is the flexibility in selecting different types of architectures, such as systems, software, and enterprise. This raises questions about how elaboration approaches may vary depending on the type of architecture at hand, and how these differences should be addressed.

The relationship with other processes is also an important topic of discussion. The Architecture Elaboration process highlights coordination with other processes, such as the Architecture Conceptualization process, prompting debates on how these processes interrelate and how information can be effectively shared among them.

The need for continuous evaluation and control during the elaboration process raises discussions about the appropriate criteria for assessing the quality of an architecture description and how to address deviations or issues identified during elaboration.

Lastly, the importance of training and competency of teams involved in the elaboration process is a relevant theme. This generates debates on how to ensure that teams possess the knowledge and skills necessary to effectively undertake architecture elaboration.

These discussions underscore the complexity and importance of the Architecture Elaboration process in describing software or systems architectures and emphasize the need to consider these aspects to ensure a description aligned with stakeholders' needs.

3.4 Summary

Standard 42020 establishes specific guidelines for the documents elaboration of architectures through the Architecture Elaboration process, detailing the activities and tasks involved in creating Architecture Descriptions suitable for their specific purposes. It emphasizes the importance of consistency, correctness, and alignment with requirements and design characteristics, as well as the relevance of change management and configuration control throughout the architecture's lifecycle.

To write these Architecture Descriptions, standard 42010 establishes the general principles and requirements for describing architectures, defining key elements such as stakeholders, viewpoints, model kinds, views, models, aspects, perspectives, and the architecture description itself, serving as a fundamental basis for standardizing the Architecture Elaboration process.

Although both standards are closely related and share essential concepts, it is important to note that they serve different purposes. While 42010 sets out general principles for describing architectures, 42020 provides detailed guidance on implementing these principles in practice.

The following chapter focuses on creating detailed best practice guidelines for the application of the Architecture Elaboration process, jointly with principles and requirements for describing architectures of the 42010 standard. This includes a step-by-step process models and concrete examples. Thus, a proposed process builds upon the Architecture Elaboration process, addressing gaps related to the "why" and "how". Through a modular approach, procedures, definitions, and methods are defined and used to support the realization of the task, ensuring flexibility and adaptability across diverse organizational and project contexts.

Thus, the tailor-made process, described in the following chapter, covers the specific complexities of software or system architecture documentation and standards raised so far. The proposed process is positioned to bridge gaps and provide granularity and comprehensibility to describe or document software architecture(s), catering to the evolving needs of the Software Architecture community.

4

A Process for the Clause 10 (Architecture Elaboration) of ISO/IEC/IEEE 42020

In contemporary practice, the description or documentation of software architecture continues to rely on models comprising views representing stakeholders' perspectives, viewpoints, and other elements consolidated in one or more documents. What sets the current approach apart is the support provided by 42020, which establishes a formal architecture elaboration clause (KRUCHTEN, 1995; IEEE, 2019).

The Clause 10 (Architecture Elaboration) of 42020 underscores the importance of aligning the architecture with other relevant architectures, tracking changes to the architecture description, and providing architecture elaboration information to intended users, playing a critical role in enhancing the architecture's quality, usability, and alignment with stakeholder concerns, contributing to the success of architecture projects and their intended objectives.

The use of the Clause 10 to create a process for documenting software architecture is a proactive and strategic investment. It facilitates effective communication, reduces the learning curve for new team members, supports ongoing maintenance and evolution, mitigates risks, and preserves institutional knowledge as shown in Table 4. A well-documented software architecture ultimately contributes to the long-term success and sustainability of a software project.

Reason	**Why**
Knowledge Transfer and Onboarding	New team members or developers joining a project need a reliable source of information to understand the software architecture. Proper documentation ensures a smooth onboarding process, reducing the learning curve for newcomers.

Maintenance and Evolution	Software is not static; it evolves. Changes, updates, and enhancements are inevitable. Documentation provides a roadmap for maintenance, making it easier for developers to understand the existing architecture, identify potential areas for improvement, and implement changes without introducing errors.
Collaboration and Team Communication	In a collaborative development environment, multiple team members need to work together. Documentation serves as a common reference point, fostering effective communication among team members. It ensures that everyone is on the same page regarding the overall design and structure of the software.
Risk Mitigation	Clear documentation helps to identify potential risks and challenges early in the development process. By documenting design decisions, dependencies, and potential pitfalls, teams can proactively address issues and mitigate risks before they impact the project.
Knowledge Preservation	Over time, team members may either leave the team or be reassigned to different projects. Documentation serves as a repository for the collective wisdom and insights of the team, ensuring their preservation over time. Such documentation ensures that the institutional knowledge about the software architecture is not lost when individuals transition in and out of the project.
Compliance and Standards	Many industries and organizations have specific standards and compliance requirements. Properly documented software architecture helps ensure that the system aligns with these standards, making it easier to pass audits and comply with regulatory requirements.

Table 4 – Going deeper into why creating a process for documenting software architecture

The Architecture Elaboration process outlines what needs to be done (activities and tasks), but it lacks information on why it needs to be done, who should do it, and how it will be carried out. The reason for this separation is to ensure the clarity and simplicity of the process document itself. For each of these aspects, supplementary information is provided in additional supporting documents and resources, which serve as a comprehensive guide or reference.

This dissertation takes advantage of this gap and defines aspects, including specific methodologies and techniques (the “why” and “how”). Subtasks are progressively elevated to tasks of the seven activities, and a novel process is simulated, thereby establishing the fundamental flow depicted in Figure 13.

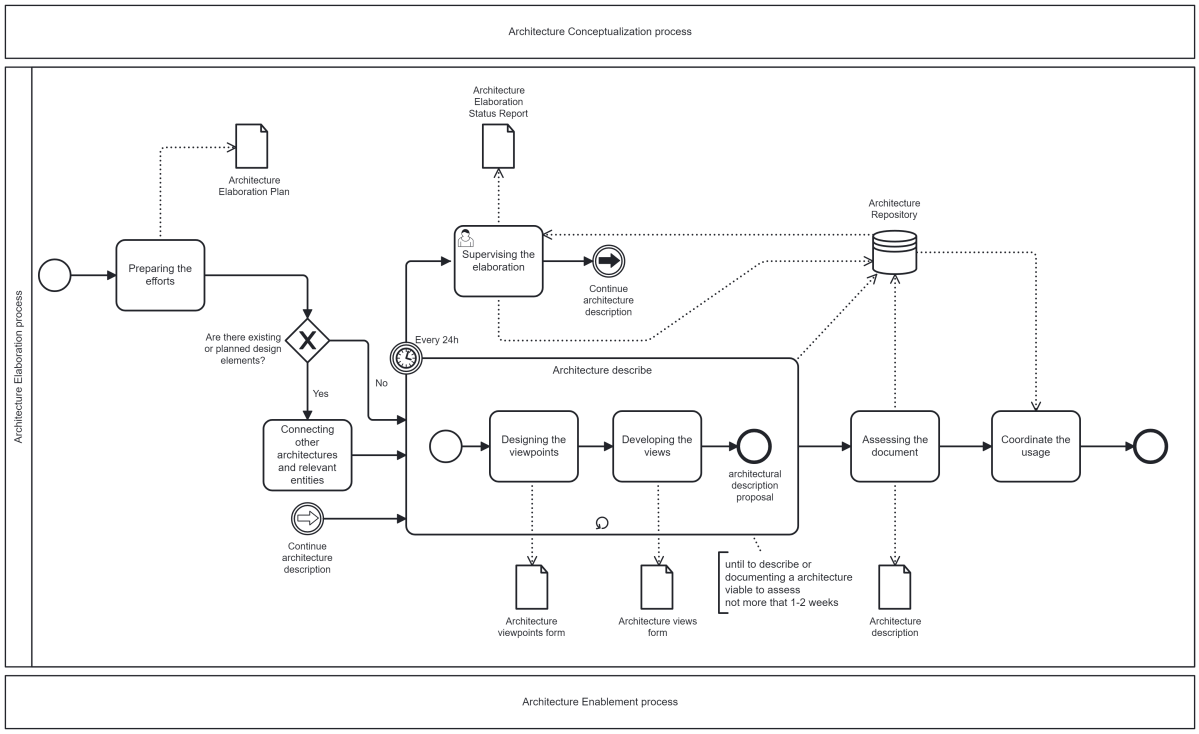


Figure 13 – Process model

4.1 Tailored Process Details

As outlined in Table 5, the process described in this dissertation commences with the utilization of the terminology delineated by Kneuper, which delineates the procedures to be carried out. Table 5 provides an overview of the process, specifying the list of activities and procedures. A procedure is a methodical approach to the execution of an activity or process (IEEE, 2023). This is consistent with the notion that processes can be documented in a uniform manner, frequently referred to as standard operating procedures (SOP) (KNEUPER, 2018).

Step	**Activity**	**Procedure**	**Architecture Elaboration Process (42020) corresponding activities**
1	Preparing the efforts	Identify the need for architecture elaboration based on project requirements. Formulate a team comprising roles like architect, developers, and domain experts. Clearly define the scope of the architecture elaboration. Establish specific objectives and expected outcomes. Allocate responsibilities to team members and roles. Develop a detailed plan outlining tasks, activities, and timelines.	Prepare for and plan the architecture elaboration effort

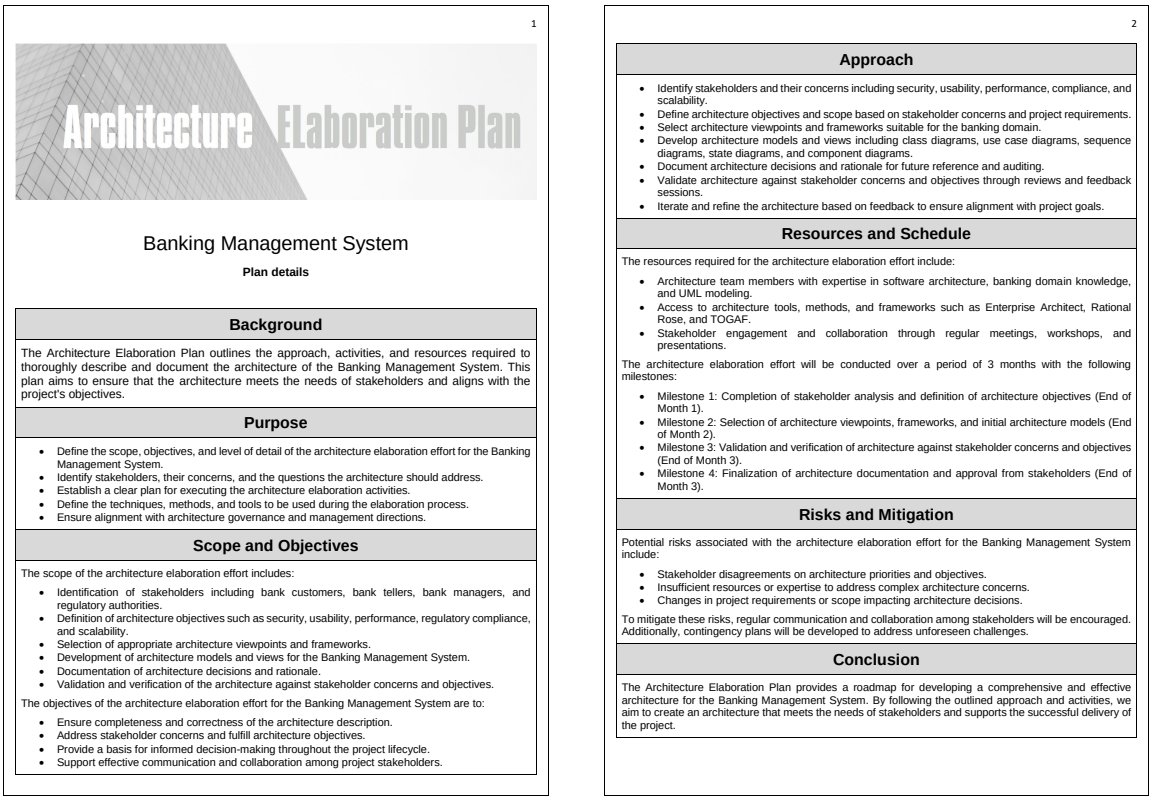
2	Designing the viewpoints	Define various architectural viewpoints based on the nature of the system. Use recognized methodologies and tools to assist in viewpoint identification.	Identify or develop architecture viewpoints
3	Developing the views	Create models and views corresponding to identified viewpoints. Utilize modelling tools and techniques to represent the architecture effectively.	Develop models and views of the architecture(s)
4	Assessing the document	Evaluate the elaborated architecture against defined objectives. Conduct reviews and assessments to validate the quality and completeness. Document the elaborated architecture comprehensively. Communicate the architecture to stakeholders, ensuring understanding and feedback.	Assess the architecture elaboration
5	Coordinating the usage	Coordinate with intended users for the effective utilization of the elaborated architecture. Provide necessary training and support.	Coordinate use of elaborated architecture by intended users
6	Connecting other architectures and relevant entities	Establish connections and relationships with other relevant architectures and entities. Ensure alignment with organizational standards and external systems.	Relate the architecture to other architectures and relevant affected entities
7	Supervising the elaboration	Collect and analyze system requirements. Identify any constraints, such as budget, technology, or time limitations.	Monitor, assess, and control the architecture elaboration activities

Table 5 – List of activities

4.1.1 Step 1 – Preparing the efforts

For this step, the organization, or team, undertakes preparatory and essential planning activities for the thorough elaboration of the software architecture. The original activity, as proposed by the Architecture Elaboration clause, delineates a total of thirty tasks. The tailored process presented herein has been streamlined to five steps, with the addition of sub-steps, methods, and tools. These modifications have been strategically incorporated to empower organizations in the efficient preparation, comprehension, and planning of software architecture elaboration initiatives. The primary objective is to guarantee that the resulting architectural description is seamlessly aligned with the intended uses and objectives.

Figure 14 depicts a practical example of the work product called Architecture Elaboration Plan (AEP),



1

Architecture Elaboration Plan

Banking Management System

Plan details

Background

The Architecture Elaboration Plan outlines the approach, activities, and resources required to thoroughly describe and document the architecture of the Banking Management System. This plan aims to ensure that the architecture meets the needs of stakeholders and aligns with the project's objectives.

Purpose

- Define the scope, objectives, and level of detail of the architecture elaboration effort for the Banking Management System.
- Identify stakeholders, their concerns, and the questions the architecture should address.
- Establish a clear plan for executing the architecture elaboration activities.
- Define the techniques, methods, and tools to be used during the elaboration process.
- Ensure alignment with architecture governance and management directions.

Scope and Objectives

The scope of the architecture elaboration effort includes:

- Identification of stakeholders including bank customers, bank tellers, bank managers, and regulatory authorities.
- Definition of architecture objectives such as security, usability, performance, regulatory compliance, and scalability.
- Selection of appropriate architecture viewpoints and frameworks.
- Development of architecture models and views for the Banking Management System.
- Documentation of architecture decisions and rationale.
- Validation and verification of the architecture against stakeholder concerns and objectives.

The objectives of the architecture elaboration effort for the Banking Management System are to:

- Ensure completeness and correctness of the architecture description.
- Address stakeholder concerns and fulfill architecture objectives.
- Provide a basis for informed decision-making throughout the project lifecycle.
- Support effective communication and collaboration among project stakeholders.

2

Approach

- Identify stakeholders and their concerns including security, usability, performance, compliance, and scalability.
- Define architecture objectives and scope based on stakeholder concerns and project requirements.
- Select architecture viewpoints and frameworks suitable for the banking domain.
- Develop architecture models and views including class diagrams, use case diagrams, sequence diagrams, state diagrams, and component diagrams.
- Document architecture decisions and rationale for future reference and auditing.
- Validate architecture against stakeholder concerns and objectives through reviews and feedback sessions.
- Iterate and refine the architecture based on feedback to ensure alignment with project goals.

Resources and Schedule

The resources required for the architecture elaboration effort include:

- Architecture team members with expertise in software architecture, banking domain knowledge, and UML modeling.
- Access to architecture tools, methods, and frameworks such as Enterprise Architect, Rational Rose, and TOGAF.
- Stakeholder engagement and collaboration through regular meetings, workshops, and presentations.

The architecture elaboration effort will be conducted over a period of 3 months with the following milestones:

- Milestone 1: Completion of stakeholder analysis and definition of architecture objectives (End of Month 1).
- Milestone 2: Selection of architecture viewpoints, frameworks, and initial architecture models (End of Month 2).
- Milestone 3: Validation and verification of architecture against stakeholder concerns and objectives (End of Month 3).
- Milestone 4: Finalization of architecture documentation and approval from stakeholders (End of Month 3).

Risks and Mitigation

Potential risks associated with the architecture elaboration effort for the Banking Management System include:

- Stakeholder disagreements on architecture priorities and objectives.
- Insufficient resources or expertise to address complex architecture concerns.
- Changes in project requirements or scope impacting architecture decisions.

To mitigate these risks, regular communication and collaboration among stakeholders will be encouraged. Additionally, contingency plans will be developed to address unforeseen challenges.

Conclusion

The Architecture Elaboration Plan provides a roadmap for developing a comprehensive and effective architecture for the Banking Management System. By following the outlined approach and activities, we aim to create an architecture that meets the needs of stakeholders and supports the successful delivery of the project.

Figure 14 – Architecture Elaboration Plan Example

The first step encompasses several pivotal sub-steps, each of which contributes to comprehensive readiness and strategic planning. The step and sub-steps are explored in Appendix A.

4.1.2 Step 2 – Designing the viewpoints

In this step, involves the identification, adaptation, or creation of specific perspectives or viewpoints through which the architecture of a system or entity can be comprehensively understood and communicated. It involves identifying relevant stakeholders, and their concerns, and tailoring viewpoints to address those concerns. These viewpoints provide lenses through which different stakeholders can understand and analyze the architecture, promoting a comprehensive and well-rounded view that accommodates diverse perspectives. Based on the viewpoint catalog (ROZANSKI; WOODS, 2012), this dissertation considers the “Context Viewpoint” to elaborate a simple example using a simplified UML model for a banking system (Figures 15 and 16).

Each viewpoint addresses specific concerns of stakeholders and provides insights into different aspects of the banking system architecture. Together, they offer a comprehensive understanding of the system’s design and functionality from various perspectives.

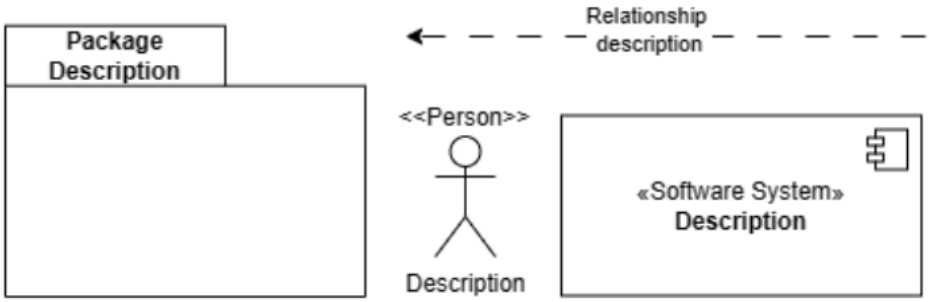
Context viewpoint	
Overview:	Describes the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts)
Concerns:	Regulatory compliance (Meeting legal and regulatory requirements)
Stakeholders:	Bank customers, bank tellers, bank managers, regulatory authorities, and developers
Stakeholder perspectives:	Regulation (The ability of the system to conform to local and international laws, quasi legal regulations, company policies, and other rules and standards).
Problem and pitfalls:	• Missing or incorrect internal and external entities • Inappropriate level of detail
Applicability:	All systems
Views:	Context view
Models:	Context model (The diagram will be illustrated using UML with the appropriate use of packages, components, and stereotypes)  *A - Metamodel related to the specification of a model kind*

Figure 15 – Simple example of the architecture viewpoint 1

The second step encompasses several pivotal sub-steps, each of which contributes to the comprehensive viewpoints. The step and sub-steps are explored in Appendix B.

4.1.3 Step 3 – Developing the views

For this step, the architecture is translated into concrete views. It encompasses the creation of diagrams, charts, and other visual representations that convey different facets of the architecture. The views serve as communication tools, aiding stakeholders in understanding complex architectural structures, relationships, and behaviors (Figures 17 and 18).

The third step encompasses several pivotal sub-steps, each of which contributes to the comprehensive models and views of the software architecture. The step and sub-steps are explored in Appendix C.

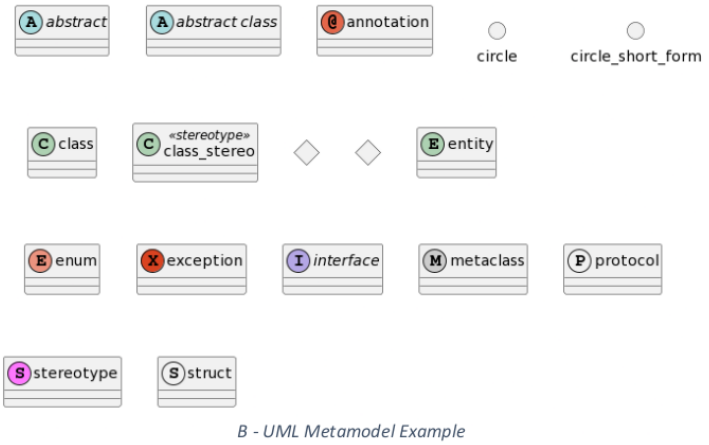
Development viewpoint	
Overview:	Definition of the viewpoint based on the bibliography. For example, the use of the development viewpoint defined on the Kruchten "4+1" viewpoint catalog.
Concerns:	Matter of relevance or importance to a stakeholder (conform the ISO/IEC/IEEE 42010).
Stakeholders:	Role, position, individual, organization, or classes thereof, having an interest, right, share, or claim, in an entity of interest (conform the ISO/IEC/IEEE 42010).
Stakeholder perspectives:	A listing of any stakeholder perspectives associated with this architecture viewpoint (conform the ISO/IEC/IEEE 42010).
Problem and pitfalls:	The problem framed by viewpoint.
Applicability:	Software systems affected.
Views:	View or view list based on the bibliography.
Models:	Category of model distinguished by its key characteristics and modelling conventions (conform the ISO/IEC/IEEE 42010). 

Figure 16 – Simple example of the architecture viewpoint 2

4.1.4 Step 4 – Assessing the document (Architecture Description)

In this step, focuses on evaluating the completeness, consistency, and quality of the proposed architecture. It entails systematically reviewing architecture views and models against predefined objectives, ensuring alignment with the intended conceptualization. The assessment process identifies discrepancies, validates the architecture against its conceptual intent, and addresses any gaps or inconsistencies, ultimately enhancing the overall robustness of the architecture.

The fourth step encompasses several pivotal sub-steps, each of which contributes to the well-assess of the architecture description. The step and sub-steps are explored in Appendix D.

4.1.5 Step 5 – Coordinating the usage

The objective of this step is to facilitate the effective utilization of the proposed architecture by its intended users. To achieve this, it is necessary to identify user needs, prepare tailored

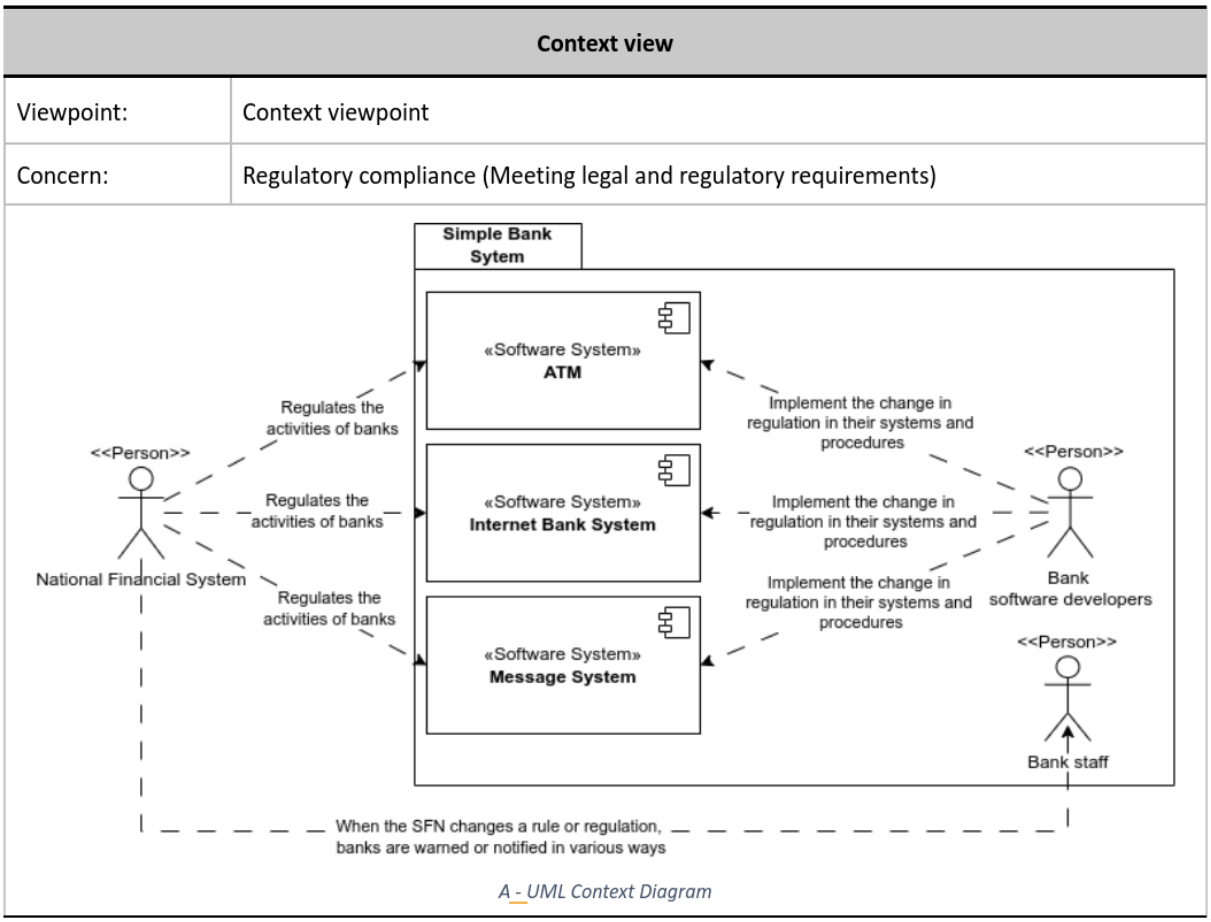


Figure 17 – Simple example of the architecture view 1

information, and maintain ongoing support. The coordination ensures that the architecture meets the diverse requirements of stakeholders, providing clarity, accessibility, and adaptability throughout its lifecycle. Communication, feedback mechanisms, and continuous improvement are integral components of this coordination effort.

The fifth step encompasses several pivotal sub-steps, each of which contributes to the efficient coordinate use. The step and sub-steps are explored in Appendix E.

4.1.6 Step 6 – Connecting other architectures and relevant entities

In this step, the objective is to establish connections between the proposed architecture and external entities, such as other architectures, systems, or organizational elements. The goal is to identify dependencies, potential reuse opportunities, and impacts on related entities. By mapping these relationships, the architecture team ensures a holistic understanding and integration with the broader context.

The sixth step encompasses several pivotal sub-steps, each of which contributes to connecting existing architectures and relevant entities. The step and sub-steps are explored in Appendix F.

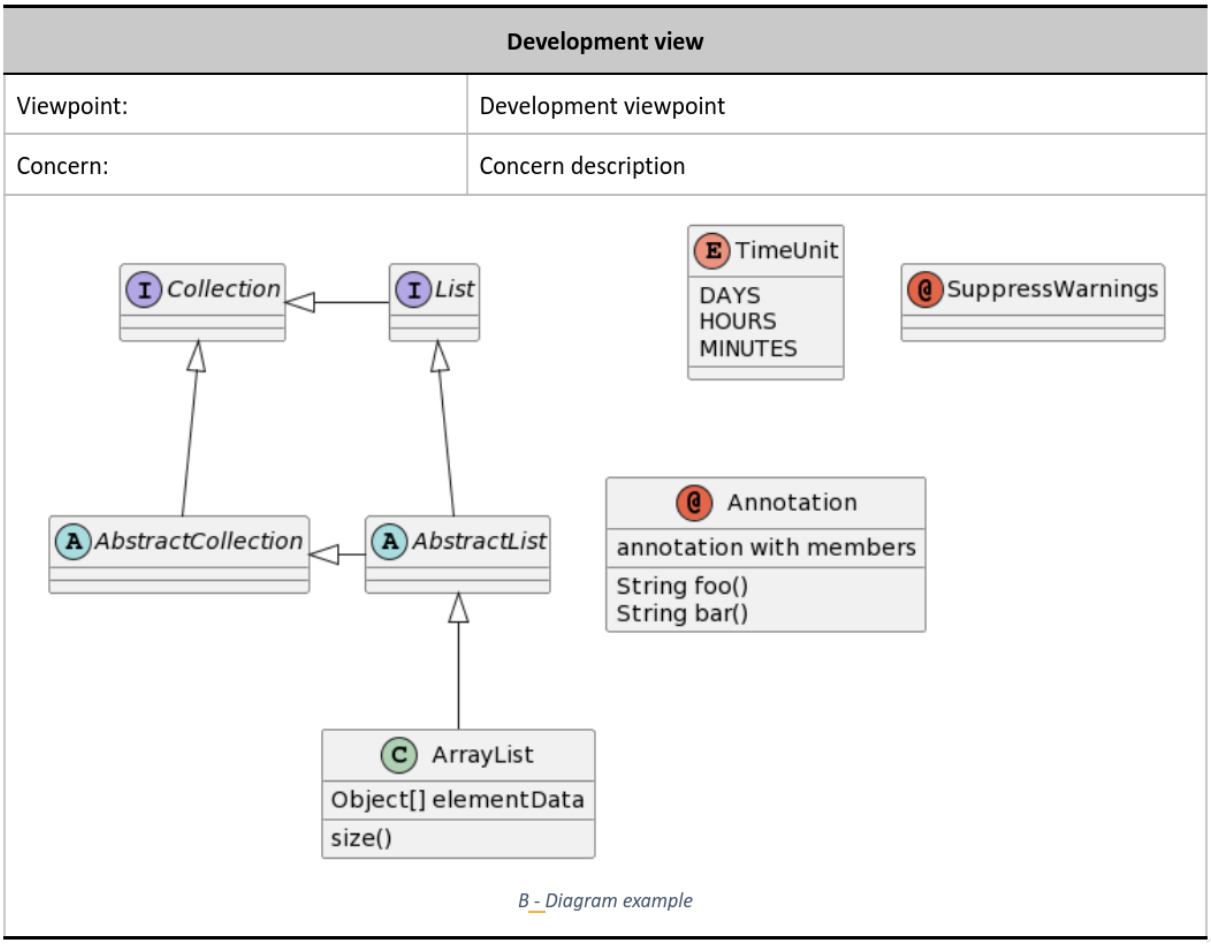


Figure 18 – Simple example of the architecture view 2

4.1.7 Step 7 – Supervising the elaboration

In this step, continuous oversight is applied to the ongoing architecture elaboration efforts. It involves systematically evaluating progress, assessing the quality of work, and ensuring adherence to the established plan. Monitoring and control mechanisms are implemented to identify and address issues promptly. This step facilitates the dynamic adjustment of strategies, ensuring that the architecture elaboration stays aligned with objectives and remains responsive to changing conditions.

The seventh step encompasses several pivotal sub-steps, each of which contributes to the production of well-supervised software architecture documentation. The step and sub-steps are explored in Appendix G.

4.2 Summary

This chapter explores software architecture documentation, revealing a structured process for elaborating on software architectures. It navigates through key activities such as preparing for and planning the architecture elaboration effort, designing architecture viewpoints, developing

architecture models, relating to other architectures and entities, assessing, communicating, and monitoring. Each step involves a series of sub-steps, highlighting the intricate nature of architecting complex software systems.

It goes beyond a step-by-step guide on how to document a software architecture, providing the community of potential users with educational content that aims to reduce difficulties in understanding concepts, properties, and principles related to the elaboration of software architectures. Furthermore, the detailed breakdown of activities elucidates practical guidance for executing these tasks effectively. The procedures, methods, and concepts for each step and sub-steps define critical aspects such as “what”, “why” and “how”. While the Architecture Elaboration Process defines only “what”.

The following chapter discusses the integration and development of a tool that help and facilitate the use and application of this personalized process.

5

ArchCE: A software to generate architecture documentation based on ISO/IEC/IEEE 42020

In the realm of software development, the importance of architecture documentation cannot be overstated. Software architecture documentation serves as a blueprint, guiding the design and development of systems to meet stakeholders' needs. Stakeholders, including developers, architects, and project managers, have various concerns that must be addressed to ensure the system's success. The role of the stakeholders involved in the development and use of software architectures is pivotal in this context. Stakeholders must gather and refine these requirements, develop architectural views that address these concerns, and make informed trade-offs to balance potentially conflicting requirements. Without thorough architecture documentation, most likely that all stakeholder concerns and requirements will be adequately addressed.

In Chapter 4, a comprehensive process aligning with the "Architecture Elaboration" clause of the 42020 was developed to reduce difficulties in understanding concepts, properties, and principles related to software architecture elaboration. To allow stakeholders involved in the development and use of software architectures to utilize the developed process for elaborating and managing the architecture, as well as assisting in its validation, a software tool named ArchCE was created.

The ArchCE tool has been designed with the intention of automating the generation of architecture documentation, in accordance with the steps outlined in the developed process. The objective of ArchCE is to streamline the aforementioned steps in order to facilitate the adoption of this process and to reduce the learning curve, thereby reducing the complexity and limitations of manually following the described steps. The automation of this process ensures adherence to industry standards and makes the process more accessible to users, regardless of their prior experience with architecture documentation.

The following sections will provide a detailed examination of the ArchCE tool, focusing on

its specific characteristics and capabilities. This section will examine the features, functionalities, and benefits of the ArchCE tool in the context of the documentation process.

5.1 Introducing ArchCE

ArchCE (Architecture Conceptualization and Elaboration) is a web application wherein users input information about a specific entity. ArchCE processes this input to generate various architectural work products, including models that visually represent the system's structure and detailed descriptions that elucidate its functionalities. Furthermore, it is a derivative of the ArchConcept software tool that was initially developed by Santos and Soares (2023).

Unlike the majority of tools, which range from basic diagramming tools to advanced documentation management systems (each offering distinct features and capabilities), ArchCE incorporates characteristics aligned with the developed process (IVANOV, 2024). As a result, it conforms to the requirements of 42010 and 42020.

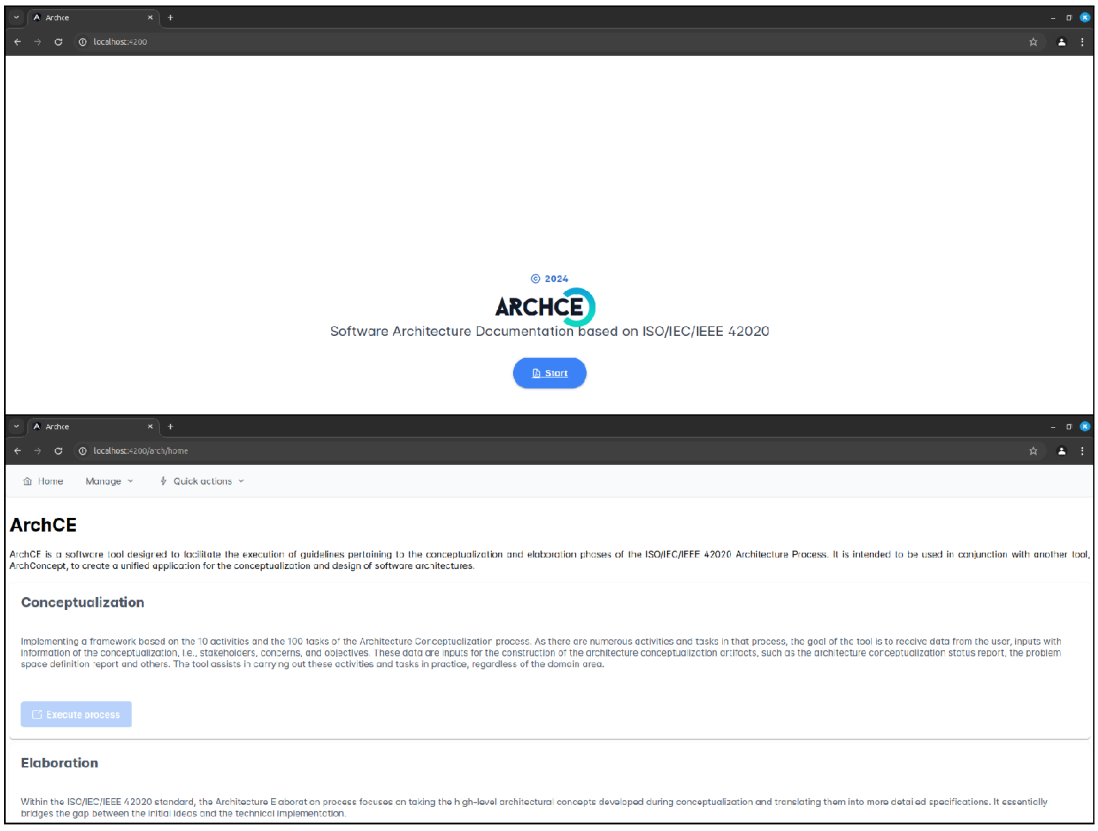


Figure 19 – Splash screen (above) and Home screen (below).

Considering the nature of ArchCE, the ArchCE project will be used as a proof of concept exemplifying the application of the developed process for the creation of the architectural description, which provides comprehensive architecture documentation. Furthermore, it delineates the manner in which minimal time can be invested in comprehending the extensive tasks associated

with Clause 10 (Architecture Elaboration) of the 42020 standard and the high-level abstract definitions by leveraging the developed process implemented throughout this application.

The ArchCE tool incorporates several features inherited from the ArchConcept tool and other features implemented to fulfill the requirements of Clause 10. However, its primary focus is on implementing process model developed. As illustrated in Figure 20, ArchCE, like the model, establishes a workflow between the four main activities of the developed process (Preparing the efforts, Designing the viewpoints, Developing the views and Assessing the document).

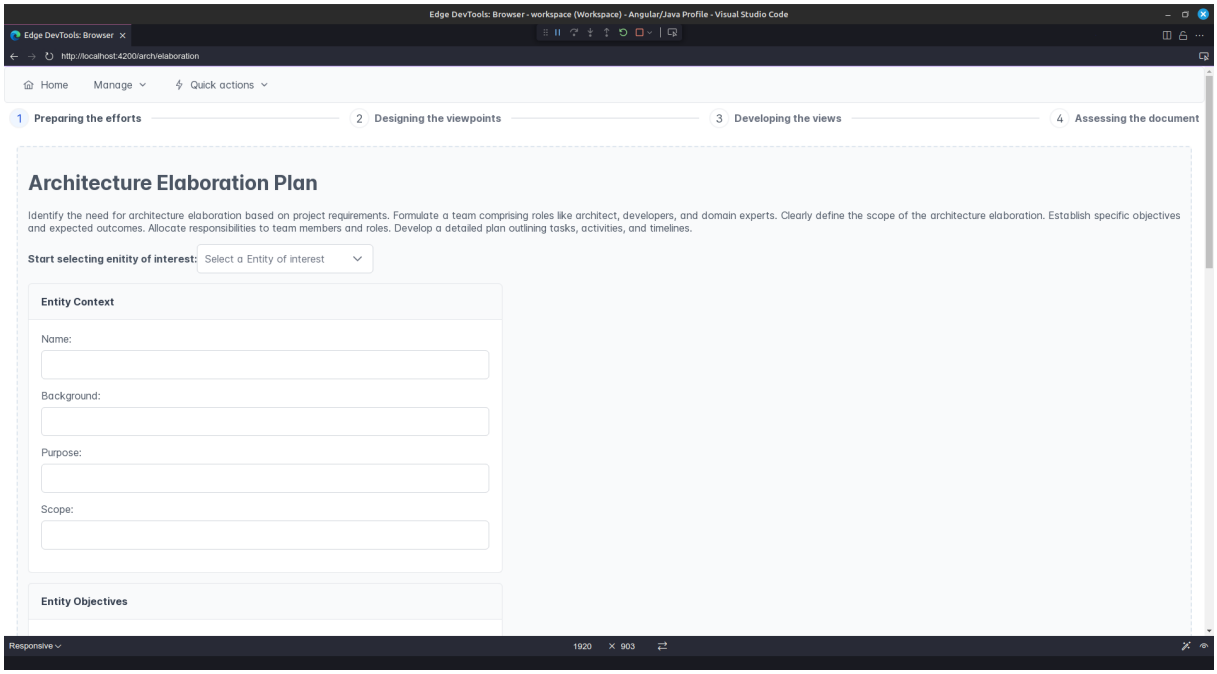


Figure 20 – Execute Architecture Elaboration Initial Screen

From here, this dissertation describes how to elaborate the architecture description of ArchCE project using the developed process beginning with preparation of efforts (step 1), including the Architecture Elaboration Plan as a work product. This step is followed by the design of viewpoints (step 2) and development of views (step 3), which includes the Architecture Viewpoint Report and Architecture View Report, respectively, as a work product. Finally, the assessment of the document (step 4) is undertaken, which includes the final work product, the Architecture Description.

5.2 ArchCE Architecture Description

The 42010 standard does not stipulate any particular format or media for recording an architecture description (IEEE, 2022). Instead, the 42010 standard specifies that an architecture description must conform to the requirements for an architecture description, an architecture description framework, an architecture description language, an architecture viewpoint, and a model kind. Furthermore, the standard specifies the best practices for documenting enterprise, system, and software architectures.

The developed process, as specified in Chapter 4, and the ArchCE tool, seek to establish a standardized format for recording an architecture description. Each step of the process outlines specific sub-steps and procedures, and the ArchCE tool further enhances this process by offering pre-built templates and checklists, automating tasks, and providing an interface to concrete documentation. This combination of a detailed process and supportive tool is designed to facilitate the systematic recording of architecture descriptions, thereby facilitating communication, understanding, and feedback among stakeholders.

With regard to the ArchCE project architecture description, according to the developed process, the stakeholders include software architects and developers with interests in a software tool. These interests are defined as concerns related to any factor that influences the system's operational context. Each system exists within a context that provides the context for all of its influencing factors. The ArchCE has been designed to function within this context, with the objective of addressing all stakeholder concerns through a comprehensive architectural description.

5.2.1 Architecture Elaboration Plan

The Architecture Elaboration Plan is a work product of the activity "Preparing the efforts". This step involves various tasks aimed at setting up the architecture elaboration process and planning information for the effort. Producing the Architecture Elaboration Plan within the context of the IEEE (2019) standard is a comprehensive framework that guides the systematic planning and execution of architecture elaboration efforts.

The initial step involves documenting the purpose, scope, and objectives of the intended stakeholders, questions, architecture elaboration approaches, techniques, methods, tools, metrics, and schedule with appropriate milestones. Figure 21 illustrates this step.

ArchCE offers stakeholders pre-built templates and checklists that streamline the completion of each step in the preparation phase. These resources are designed to collect and document all necessary information in an accurate and timely manner, without requiring significant time or effort from users. This is achieved through automation of several tasks.

Figure 22 illustrates the ArchCE Architecture Elaboration Plan, which is a document generated by the aforementioned tool. As the document is lengthy, it has been limited to two pages. To view the entire document, it can be found in Annex A.

To generate the Architecture Elaboration Plan, it is necessary to fill a number of fields. These fields allow for the input of text that extends beyond the limitations of plain text, thus enabling the application of formatting and visual elements to enhance the appearance and readability of the document. For instance, the "Scope" field was used to outline functional/nonfunctional requirements in table format. Another example is the "Approaches" field, which defines techniques, methods, tools and management directions using a topic structure based on enumeration.

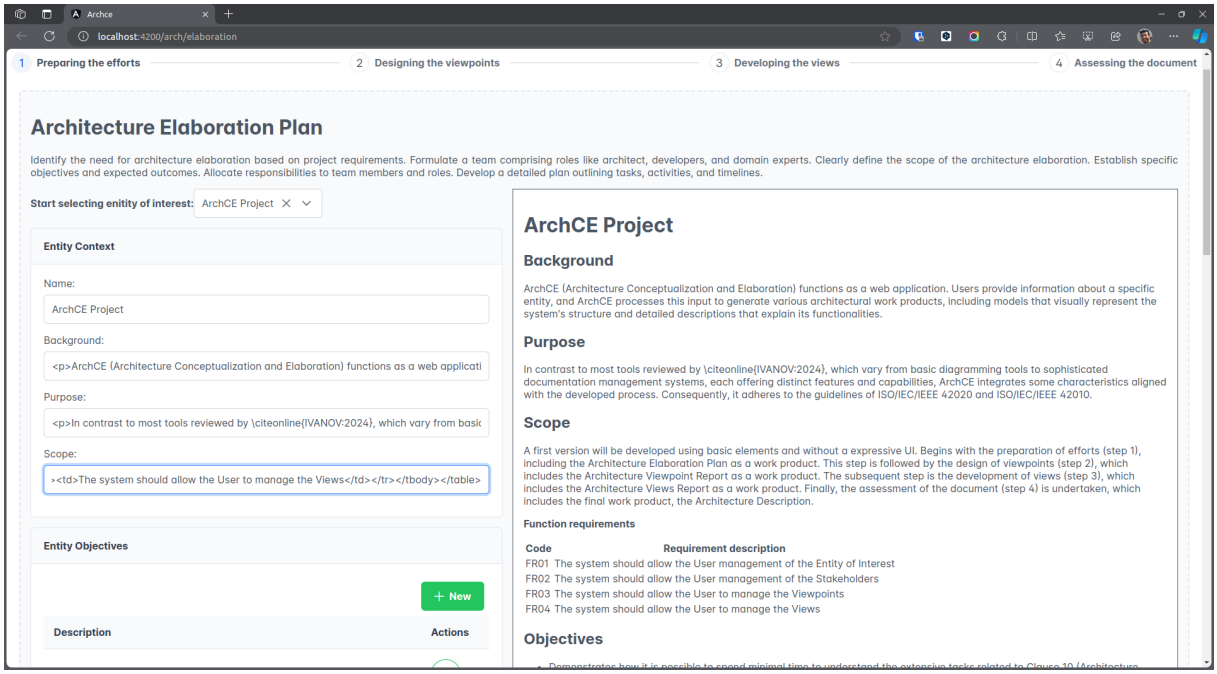


Figure 21 – Preparing efforts step

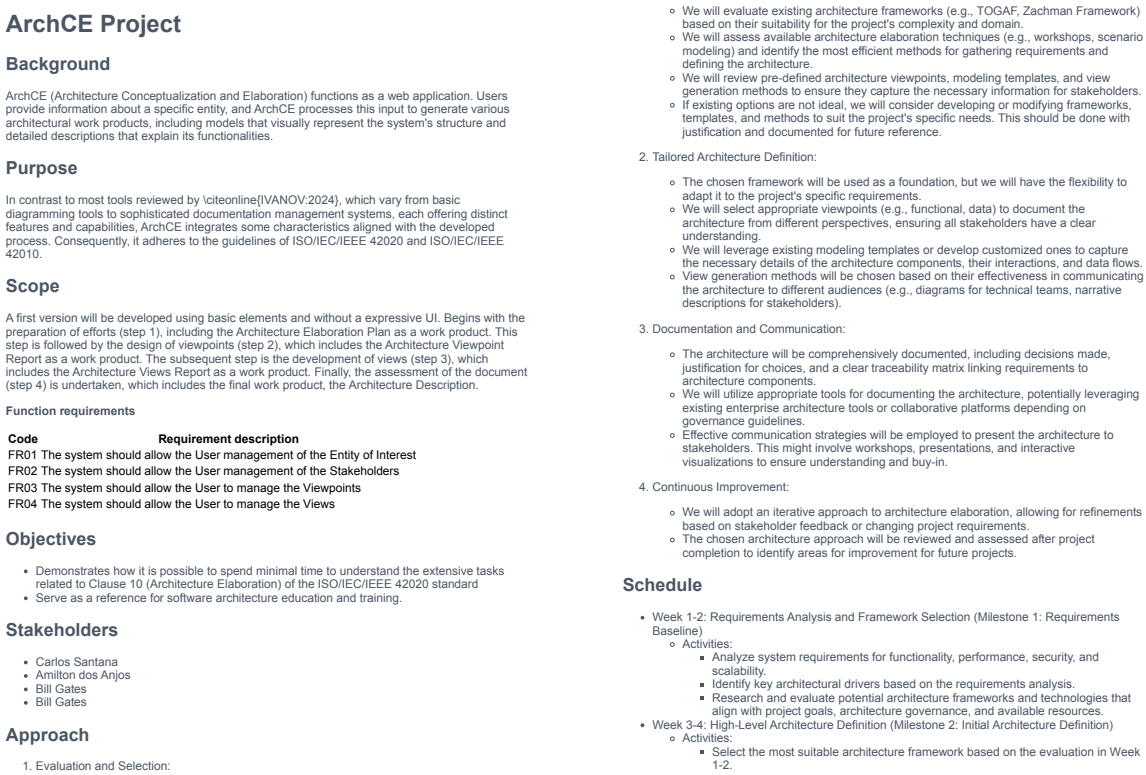


Figure 22 – ArchCE Architecture Elaboration Plan

In the case of ArchCE, taking the "framework" element as an example, the Rozanski and Woods (2012) was utilized as support to elaborate viewpoints and views, considering the concerns of the stakeholder (the developer). Further details will be provided in the next section.

The example shown here is stored by the ArchCE in the data repository for future reference, audit, training, and more. The repository can be used to facilitate understanding and ensure personnel have necessary and appropriate access to the data and information of the plan.

5.2.2 Architecture Viewpoints and Views forms

Thinking in viewpoints involves thinking in views, and the typical starting point is Philippe Kruchten's "4+1" View Model, which proposes organizing the description of software architecture into five views: Logical View, Development View, Process View, Physical View, and Scenarios (KRUCHTEN, 1995).

However, designing these views from scratch and repeatedly returning to the basics to add new definitions or create a view is a challenging task, particularly concerning stakeholder communication. The 42010 standard generalizes the idea of using views to describe software architecture and introduces the concept of a viewpoint, which serves as a template for creating views (ROZANSKI; WOODS, 2012; IEEE, 2022). Unlike Kruchten's model, the IEEE standard does not prescribe a specific set of views, allowing for flexibility in defining architecture based on different stakeholders' concerns.

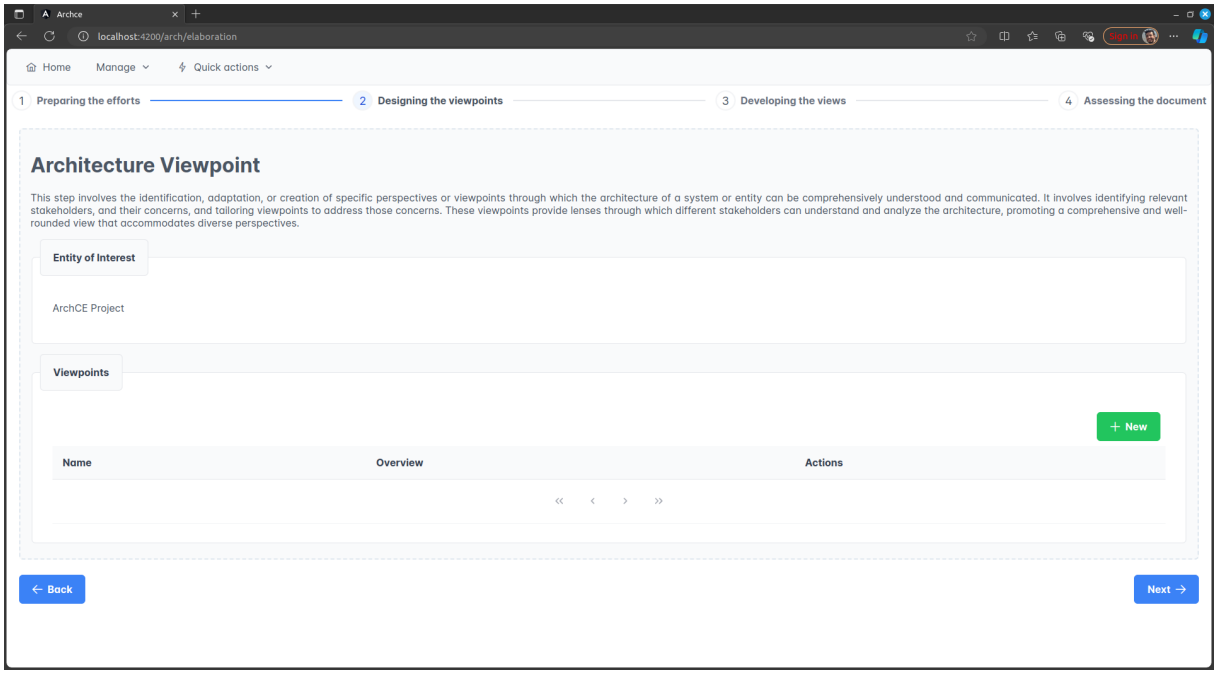


Figure 23 – Designing Viewpoints Screen

The second and third steps of the developed process, "Designing Viewpoints" (Figure 23) and "Developing the Views" (Figure 24), focus on creating a comprehensive set of architectural viewpoints and views to address different stakeholder concerns and ensure a well-rounded architectural description. According to the literature Clements et al. (2010), Rozanski and Woods (2012), Bass, Clements and Kazman (2021b), it is more common to design a viewpoint for each view, though it is possible to create a set of views for a single viewpoint.

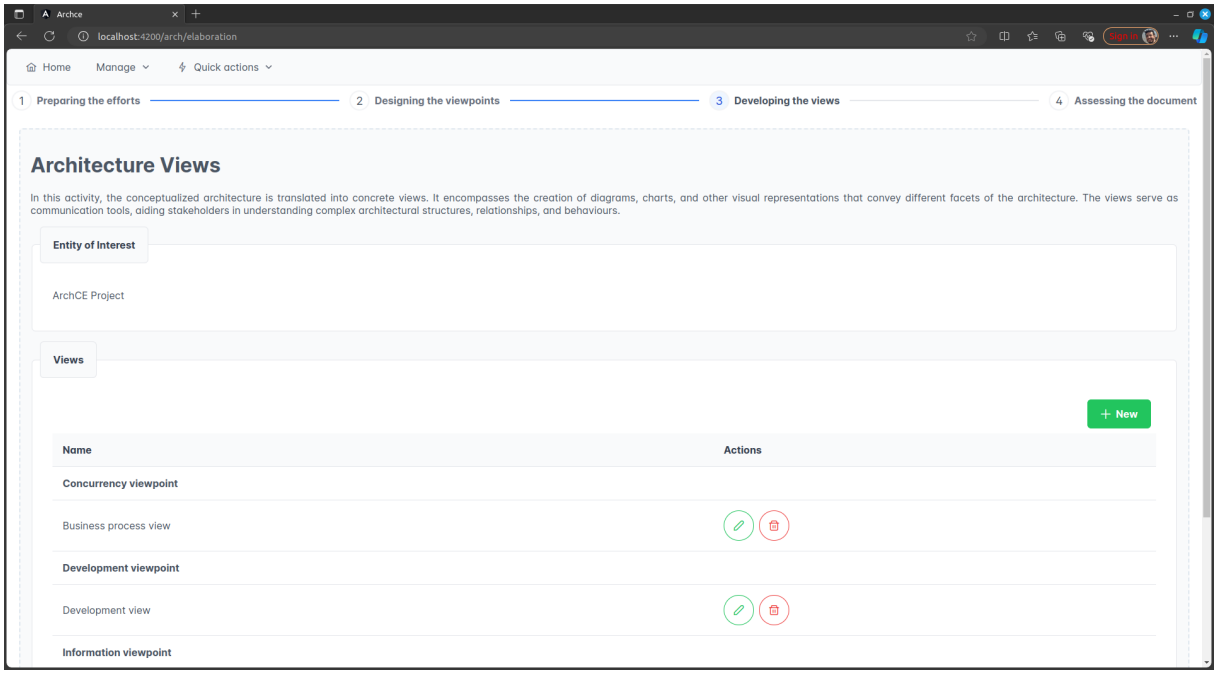


Figure 24 – Developing Views Screen

In the first step (Preparing Efforts), high-priority stakeholders were selected. For the ArchCE Project, one stakeholder was selected: the developer, whose concerns revolve around the development perspective. Rozanski and Woods (2012) present viewpoint templates that help establish a set of viewpoints.

Information viewpoint and view

Figures 25 and 26 illustrate the standardized form of viewpoint. The generated document describes how the system stores, manipulates, manages, and distributes information. The concern addressed is the structure and content of the information managed by the system, focusing on information structure modeling. The model adopted to represent this concern encompasses static information structure models (data elements and their relationships).

Concurrency viewpoint and view

Figures 27 and 28 illustrate the concurrency structure of the system, mapping functional elements to concurrency units to identify which parts can execute concurrently and how this is coordinated and controlled. The concern addressed is state management; in the case of the ArchCE Project, this involves business state management, defining valid states and transitions for core persistent information. The model adopted to represent this concern encompasses the business process model.

Development viewpoint and view

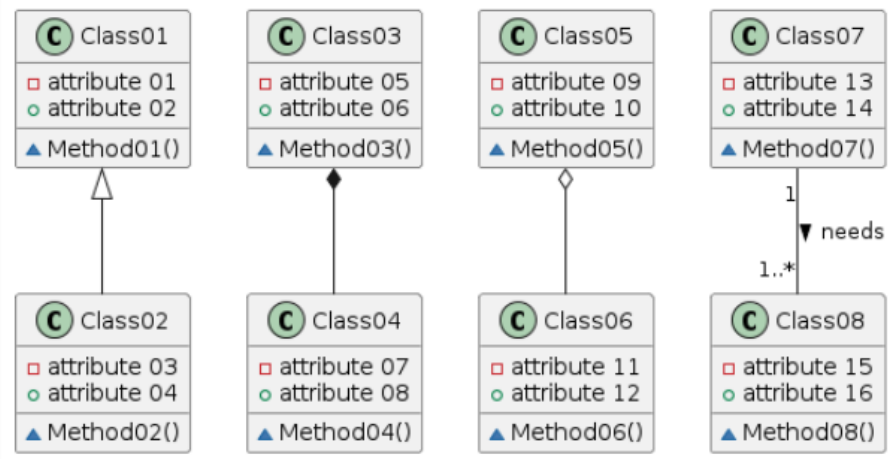
ArhCE Project	
Information viewpoint	
Overview:	Describes the way that the system stores, manipulates, manages, and distributes information
Concerns:	Information Structure and Content
Stakeholder(s):	Bill Gates
Stakeholder perspective(s):	Development
Views:	Context view
Model Kinds:	**Static Information Structure Model** The diagram will be presented using Unified Modeling Language (UML) notation, with the appropriate use of classes, relationships between classes, stereotypes, and other relevant elements. To generate the diagram, the use of PlantUML is strongly recommended. Example: 
Source:	UML2 notation; Rozanski and Woods (2012)

Figure 25 – Standardized form of information viewpoint

Figures 29 and 30, illustrate the API architecture of the ArchCE tool that supports the software development process. The concern addressed is module organization, which focuses on layer division, allowing developers to work on individual modules without unintended impacts on others. To effectively manage complex module structures, it is crucial to identify, understand, and manage dependencies between modules to prevent the system from becoming difficult and error-prone to maintain, build, and release.

The developed process is composed of seven activities, four activities are essential: Preparing the efforts, Designing the viewpoints, Developing the views, and Assessing the document. ArchCE begins with the preparation of an Architecture Elaboration Plan, outlining

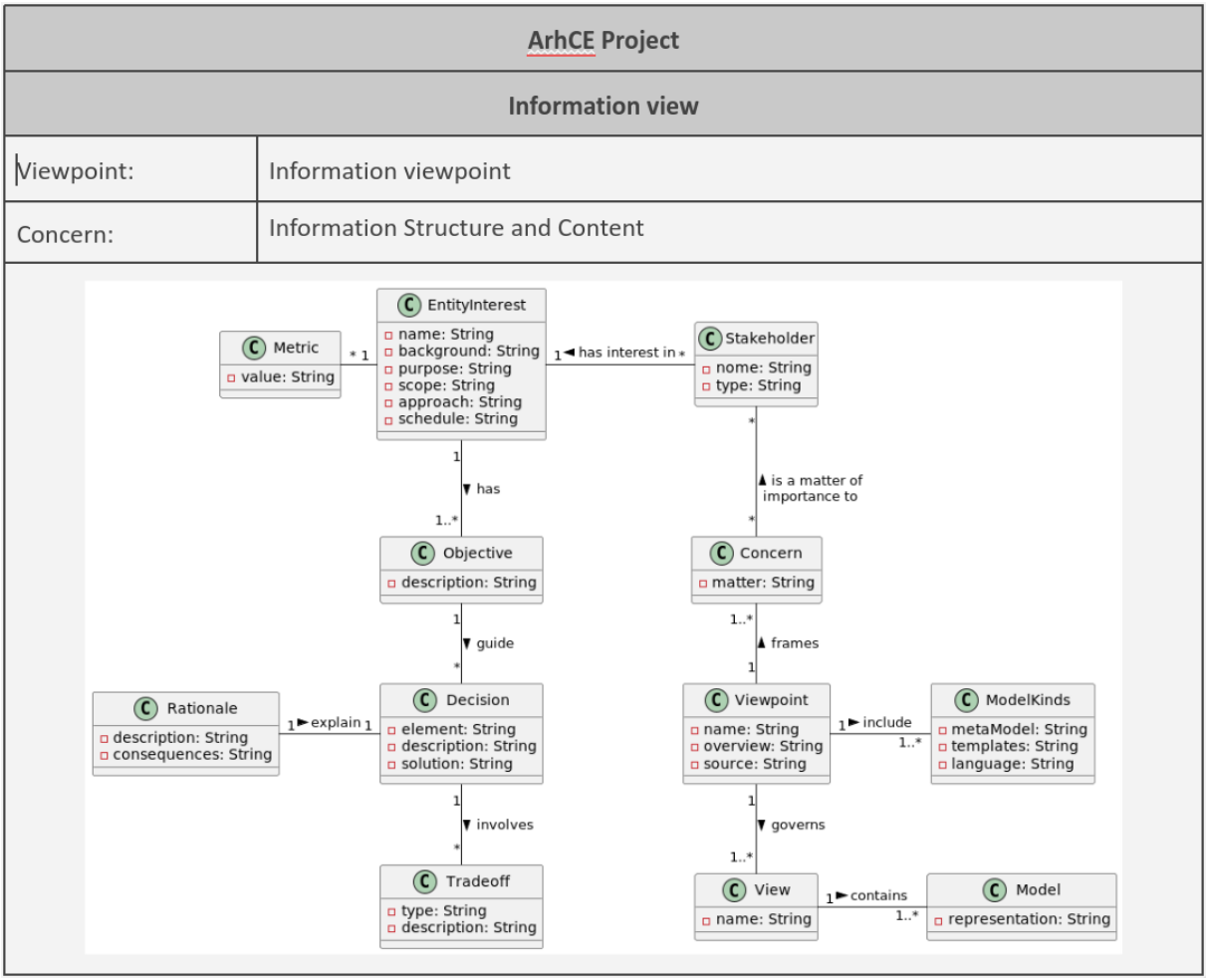


Figure 26 – Standardized form of information view

the scope, objectives, and methodologies for the architectural effort. This plan serves as a foundational document that guides subsequent activities. Following this, ArchCE supports the design of architectural viewpoints and the development of corresponding views that address stakeholder concerns comprehensively. Each viewpoint and view is structured according to predefined templates, ensuring consistency and clarity across the documentation.

The final of the four essential activities produces the Architecture Description, which is the organization, evaluation, and validation of all the other work products: Architecture Elaboration Plan, Architecture Viewpoints Forms, and Architecture Views Forms.

5.2.3 Architecture description

The name of this section, “Architecture Description”, represents the final of the four essential activities, which is the organization, evaluation, and validation of all the other work products: “Architecture Elaboration Plan”, “Architecture Viewpoints Forms”, and “Architecture Views Forms”.

Figure 31 presents the architecture validation checklist. This checklist is employed to

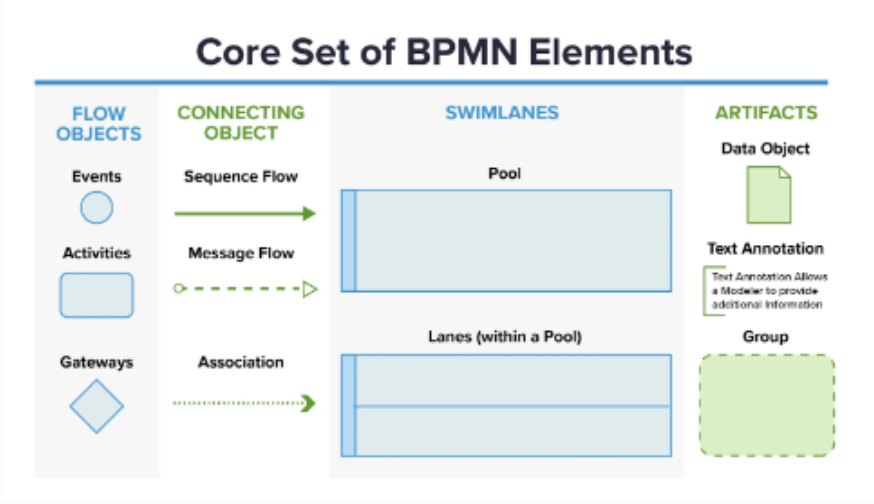
ArchCE Project	
Concurrency viewpoint	
Overview:	Describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled.
Concern(s):	Business state
Stakeholder(s):	Bill Gates
Stakeholder perspective(s):	Development
View(s):	Business Process View
Model(s):	**Business Process Model** A basic process in the business process model would normally contain the following types of entities: 
Source:	BPMN notation; Rozanski and Woods (2012)

Figure 27 – Standardized form of concurrency viewpoint

assess the completeness, consistency, and quality of an architecture document. The checklist includes items such as whether the architecture views and models have been reviewed, whether the architecture descriptions have been evaluated, and whether the suitability of the architecture description has been assessed.

The process of evaluating architectural views and models during architectural elaboration is meticulously designed at each step to ensure the coherence, effectiveness, and fidelity of the architecture. It is of the utmost importance to initially assess every architectural view and model against predefined objectives and inquiries. This is done in a systematic manner, with each view and model being reviewed and validated to identify and resolve discrepancies. This thorough

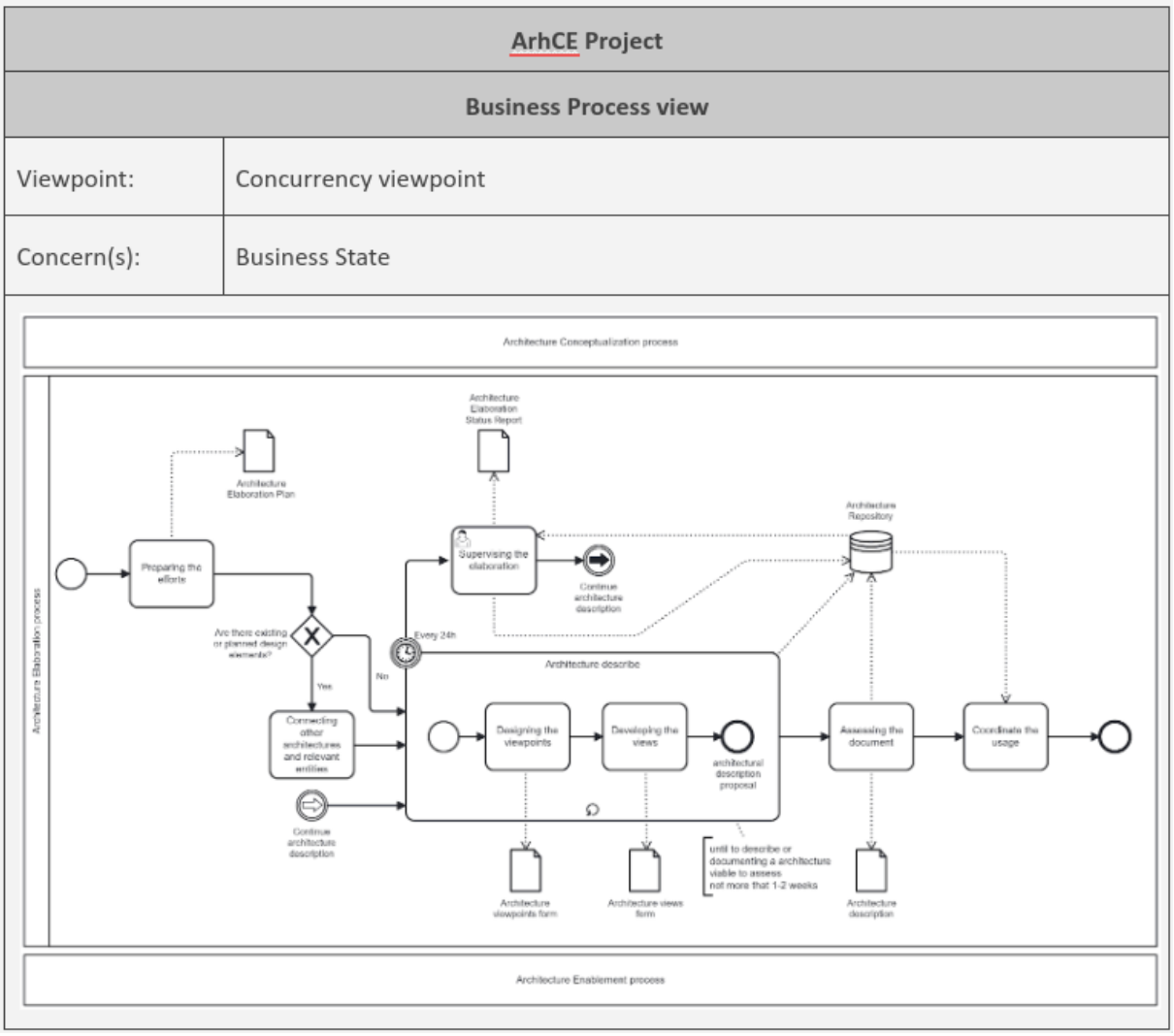


Figure 28 – Standardized form of concurrency view

evaluation phase guarantees that the envisioned architecture aligns with its actual implementation. Following this, the evaluation and verification of architecture descriptions and work products take place, ensuring that the architecture is consistent with the original vision. Comparative analysis between the elaborated and initially conceptualized architecture highlights any inconsistencies that need addressing.

5.3 ArchCE Results

This section presents the key findings and results obtained from the development, testing, and initial implementation of the ArchCE tool. The ArchCE tool was designed to support the developed process, guided by Clause 10 of the 42020. The results highlight the effectiveness of the tool in software architecture documentation and management.

ArhCE Project	
Development viewpoint	
Overview:	Describes the architecture that supports the software development process
Concern(s):	Module Organization
Stakeholder(s):	Bill Gates
Stakeholder perspective(s):	Development
View(s):	Development view
Model(s):	**Module Structure Models** The diagram is represented with a UML component diagram, using the package icon to represent a code module and dependency arrows to show intermodule dependencies. **Notation** 
Source:	UML2 Version; Rozanski and Woods (2012)

Figure 29 – Standardized form of development viewpoint

5.3.1 Effectiveness of the ArchCE Tool

The primary objective of the ArchCE tool is to facilitate the utilization of the developed process by stakeholders involved in the development and use of software architectures. The process is designed to assist stakeholders in the elaboration and management of the architecture, as well as in its validation. The effectiveness of the tool was evaluated based on the following criteria:

Adherence to Standards: The ArchCE tool successfully incorporates the principles and guidelines of Clause 10 of the 42020. It provides a structured approach to documenting

ArhCE Project	
Development view	
Viewpoint:	Development viewpoint
Concern:	Module Organization

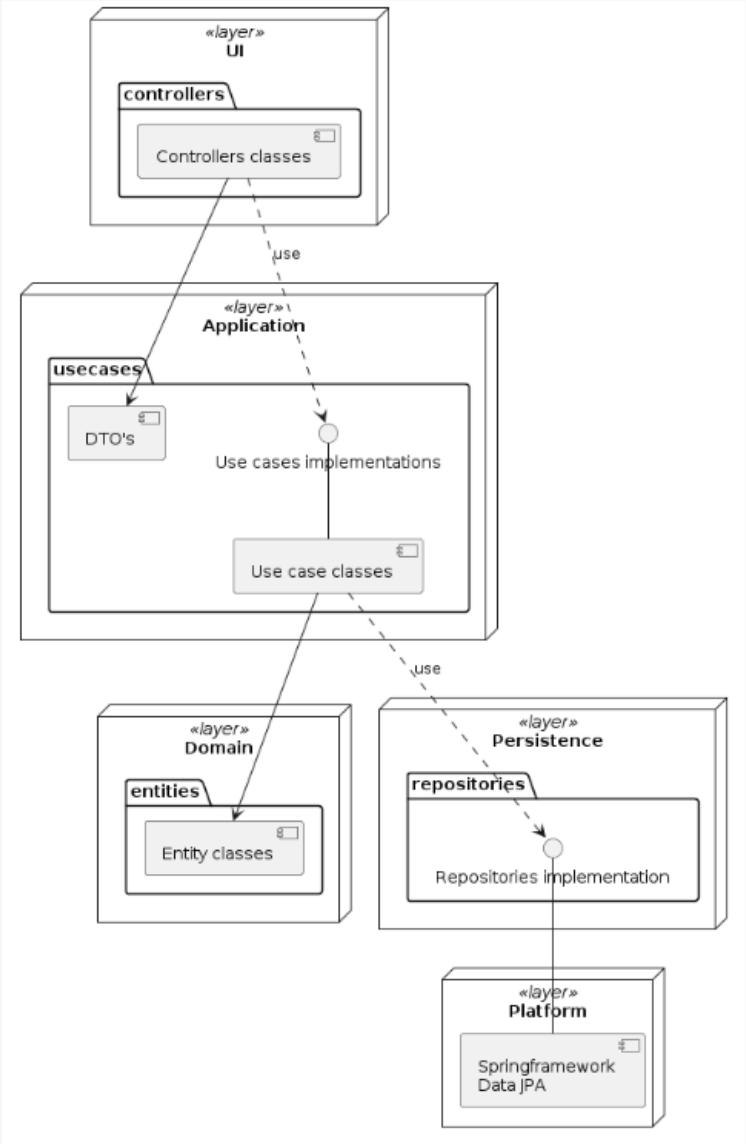


Figure 30 – Standardized form of devolopmen view

architecture viewpoints, views, and models, ensuring consistency and comprehensiveness.

Improved Documentation Quality: The tool enhances the quality of architecture documentation by offering standardized forms and templates. This reduces the likelihood of errors and omissions, leading to more accurate and complete documentation.

Facilitated Communication: By providing a clear and structured framework, the ArchCE tool improves communication among stakeholders. The standardized documentation format

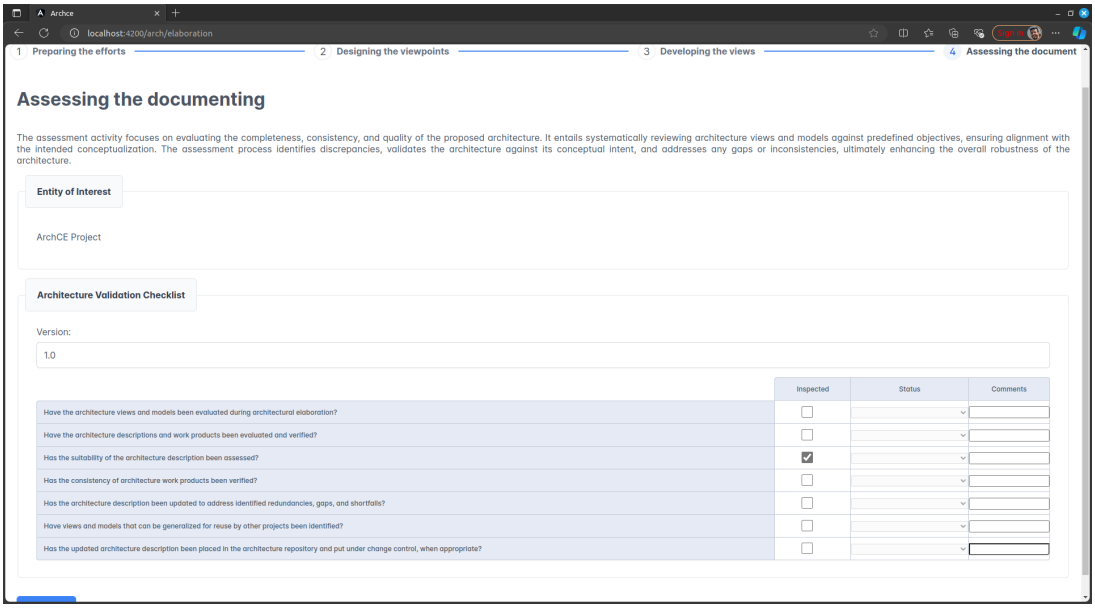


Figure 31 – Assessing the document screen

ensures that all stakeholders have a common understanding of the architecture, facilitating better decision-making and collaboration.

5.3.2 Limitations and Future Activities

While the ArchCE tool has shown significant promise, there are several limitations and future activities:

Case Studies in Real Scenarios: Further validation of the tool is required through extensive case studies in real-world scenarios. This will provide a more comprehensive understanding of the tool's effectiveness and identify any context-specific challenges.

User Feedback and Questionnaires: A structured feedback mechanism involving questionnaires and interviews with participants, researchers, and software development professionals is needed. This will help gather detailed insights into the tool's usability and impact, guiding future improvements.

Integration with Existing Tools: Enhancing the tool's compatibility with other architecture and project management tools will be a focus for future development. Which will ensure seamless integration into existing workflows and improve overall usability.

The ArchCE tool represents a significant advancement in the field of software architecture documentation. Its adherence to international standards, user-friendly interface, and practical application demonstrate its potential to improve the quality and efficiency of architecture documentation processes. Ongoing enhancements and extensive validation through real-world case studies will further solidify its position as a valuable resource for software architects and developers.

6
Conclusion

This chapter deals with the main conclusions of this Master's thesis, including: (6.1) the final conclusion of the thesis, which reiterates the issues raised by the problem and the research objectives; (6.2) the limitations of the research; and (6.3) the possibilities for future work on the subject.

6.1 Achievements and Research Outcomes

This research presents a comprehensive understanding of the elaboration of software architecture in accordance with Clause 10 of the 42020 standard. It highlights the practical and detailed application of the main activities, tasks, and subtasks developed from this clause. As evidenced by the chapters developed and in alignment with the overarching objective of this work (create a process that aims to reduce difficulties in understanding concepts, properties, and principles related to software architecture elaboration), the outcomes demonstrates that a formal process based on standards not only enhances the quality of software architecture documentation but also facilitates communication between stakeholders, enhances project management, and supports strategic decision-making.

As presented in Chapter 2, studies in the field indicate that a more comprehensive understanding of the standard can be achieved by detailing its clauses. Standardizing the elaboration of software architecture allows for the avoidance of individualizing processes that should be read in a standardized way. This issue was not only addressed in this master's thesis but also in academic events, as evidenced by Costa and Soares (2023).

Another outcome of this research is the conclusion that the use of the 42020 standard in conjunction with the 42010 standard establishes a common terminology for designing software architectures. In other words, the 42010 standard alone does not establish how to describe software architecture; rather, it describes what to describe when describing a software architecture. The

42020 defines the methodology for preparing and managing software architecture documentation. This is further elaborated upon in Clause 10, which emphasises the integration of theoretical guidelines with practical activities to enhance architectural practices. This integration is achieved by focusing on activities and tasks that take into account stakeholder concerns. These issues have been explored in two forthcoming articles: (1) A Process for the Clause Architecture Elaboration of ISO/IEC/IEEE 42020 and (2) A Conceptual Review of the Relationship between ISO/IEC/IEEE 42010 and ISO/IEC/IEEE 42020.

Finally, chapters 4 and 2 of the study described that it is possible to implement the theoretical concepts and activities discussed throughout the research in practice. The implementation of a tool responsible for helping to manage and document architectural decisions, taking stakeholder requirements and concerns as its main source and aligned with standards, is an example of this. The study highlighted the importance of procedures to achieve results such as comprehensibility and ease of use, as well as improvements to the artifact. This reinforces the findings of previous studies by Amalfitano, Luca and Fasolino (2023), Ernst and Robillard (2023), and Ivanov (2024)

It can be concluded that one of the principal findings and contributions of this research is the development of a process that encapsulates the principles of Clause 10 of 42020. This process was created to prove the main research hypothesis, which states that by outlining steps, the reference process proposed by 42020 not only facilitates the creation of software architectures, but also improves communication between stakeholders. The structured approach serves to mitigate the risk of overlooking critical architectural issues and contributes to the communication of software projects.

Moreover, the introduction of the ArchCE software tool as a practical implementation of the developed process represents a notable achievement. This tool exemplifies the tenets of the research, offering a tangible foundation for architects to operationalize the developed process in tangible contexts. The ArchCE tool facilitates the documentation of architectural decisions, ensuring alignment with the standard, and enables the evolution of the architecture and contribute to the software development lifecycle. The practical utility of the research reinforces its commitment to bridging the gap between theoretical models and practical application.

6.2 The limitations of the research

Although the research has achieved its objectives and made a valuable contribution, it is important to acknowledge and critically examine the weaknesses identified throughout the study. The aforementioned weaknesses, if addressed, could potentially enhance the outcomes of the research. The identified weaknesses are as follows:

- Although the process is complete, it is nevertheless considered to be excessively extensive. The extensive nature of the developed process can be advantageous in the management

of complex systems, as it provides comprehensive guidance and ensures comprehensive coverage of all necessary steps. Nevertheless, in agile development environments, teams frequently prioritize efficiency and agility. An overly detailed structure may impede the ability to make prompt decisions and adapt to changing circumstances.

- The development of a software tool (ArchCE) provides a practical demonstration of the theoretical concepts that underpin the research, thereby increasing its relevance. Nevertheless, the assessment of ArchCE may be insufficient in terms of its depth, particularly with respect to its usability, the challenges associated with its adoption, and its effectiveness in actual operational contexts. Although these aspects were included in the research scope, they were not fully explored due to prioritization constraints. This leaves room for subsequent research that involves comprehensive usability testing of the ArchCE tool in various real-world scenarios to gather detailed feedback. This would entail the evaluation of user experience, satisfaction, the learning curve, and adoption barriers among different stakeholder groups.

- Furthermore, the research may entail comparative analyses, contrasting ArchCE with alternative architectural documentation tools to elucidate its distinctive advantages and potential avenues for enhancement.

- Longitudinal studies could also be conducted to observe the tool's effectiveness over time in various projects, assessing its impact on the quality of the architecture and documentation practices. Based on user feedback, improvements to the tool could be developed and tested for their effectiveness in improving usability. Additionally, exploring integration of the tool with additional software architecture standards and methodologies could broaden its applicability. In-depth case studies of organizations implementing the tool could provide valuable insights into the benefits and challenges faced.

6.3 Potential avenues for future research

The findings of this study describes that the practice of working with software architecture and processes necessitates a willingness to undergo constant evolution. However, evolution does not imply a lack of structure or disregard for established processes and standards that can facilitate and qualify the results of a well-designed software architecture. It is therefore necessary to conclude this endeavor with the understanding that the universe of issues and situations explored here represents only a small fraction of the vast array of possibilities that exist. It is therefore necessary to identify further avenues for investigation:

- This research has the potential to focus on optimizing comprehensive software architecture processes in order to achieve a balance between rigor and practical usability in dynamic environments. One avenue for further research would be to examine methods for streamlining

these processes, ensuring that they remain effective and adaptable without overwhelming users.

- The study could also assess the impact of process complexity on team efficiency, decision-making, and stakeholder engagement. This would provide valuable insights into best practices for software architecture in agile settings. Such research would contribute significantly to the improvement of the application of standards in real-world scenarios.
- Another potential area of study is the use of Large Language Models (LLMs) in requirements analysis, where they can analyze and synthesize stakeholder contributions within ArchCE, ensuring that all requirements are captured accurately and efficiently. This would not only speed up the early stages of the architecture process, but also improve the overall quality of the outcome. In addition, LLMs can automate the generation of comprehensive documentation, reducing the time and effort required by architects and ensuring consistency throughout the project documentation. During the design phases, LLMs can provide real-time support to help architects explore design alternatives, identify potential issues, and make informed decisions. This real-time problem-solving capability can significantly improve the efficiency of the architectural design process.

6.4 Thesis-Related Publications

This section lists the publications directly derived from this thesis, detailing the dissemination of the research findings to the academic and professional communities.

Published Papers

COSTA, J. E. C.; SOARES, M. S. **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: XIX BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS. Maceió Brazil: ACM, 29 maio 2023. p. 412-418. Disponível em: <https://dl.acm.org/doi/10.1145/3592813.3592932>.

Submitted and Under review Papers

COSTA, J. E. C.; SOARES, M. S. **A Conceptual Review of the Relationship between ISO/IEC/IEEE 42010 and ISO/IEC/IEEE 42020**. Science of Computer Programming. Elsevier. 20 p. Manuscript number: SCICO-D-24-00161. Submitted in: June 2024.

COSTA, J. E. C.; SOARES, M. S. **A Process for the Clause Architecture Elaboration of ISO/IEC/IEEE 42020**. Journal of Systems and Software. Elsevier. 31 p. Manuscript number: JSSOFTWARE-D-24-00428. Submitted in: May 2024.

Software registry

As part of this research, two software tools that support the architecture elaboration process were developed and registered. These tools facilitate the practical application of the proposed methodology and are made available for public use.

1. Tool Name: ArchConcept
    - Description: This software is an experimental implementation of the Architecture Conceptualization Process as defined in the ISO/IEC/IEEE 42020 Standard. The objective of this software is to assist software architects in increasing their productivity and obtaining high-quality results by implementing the standard's recommendations for software architecture conceptualization.
    - Notification code: NIB517-2024
2. Tool Name: ArchCE
    - Description: This software serves to supplement the ArchConcept, extending the experimental implementation to encompass the Architecture Elaboration clause as defined in the ISO/IEC/IEEE 42020 Standard. The objective of this software is to assist software architects in efficiently understanding the extensive tasks related to Clause 10 and high-level abstract definitions. This is achieved by using the developed process implemented throughout this application.
    - To be submitted

Bibliography

ABOWD, G.; ALLEN, R.; GARLAN, D. Formalizing style to understand descriptions of software architecture. *ACM transactions on software engineering and methodology*, ACM, v. 4, n. 4, p. 319–364, 1995. ISSN 1049-331X. Cited on page 13.

ABRAHAMSSON, P.; BABAR, M.; KRUCHTEN, P. Agility and Architecture: Can They Coexist? *IEEE Software*, v. 27, n. 2, p. 16–22, Mar. 2010. ISSN 0740-7459. Cited on page 34.

ALDENHOVEN, C. M.; ENGELSCHALL, R. S. The beauty of software architecture. In: *2023 IEEE 20th International Conference on Software Architecture (ICSA)*. L'Aquila, Italy: IEEE, 2023. p. 117–128. ISBN 9798350397499. Cited on page 34.

AMALFITANO, D.; LUCA, M. D.; FASOLINO, A. R. Documenting Software Architecture Design in Compliance with the ISO 26262: a Practical Experience in Industry. In: *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*. L'Aquila, Italy: IEEE, 2023. p. i–xi. ISBN 978-1-66546-459-8. Available at: <https://ieeexplore.ieee.org/document/10092726/>. Cited on page 72.

BAABAD, A. et al. Software Architecture Degradation in Open Source Software: A Systematic Literature Review. *IEEE Access*, v. 8, p. 173681–173709, 2020. ISSN 2169-3536. Cited on page 35.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software architecture in practice*. 4. ed. Boston, MA: Addison Wesley, 2021. (SEI Series in Software Engineering). Cited on page 15.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software architecture in practice*. Fourth edition. Boston: Addison-Wesley, 2021. (Sei series in software engineering). ISBN 978-0-13-688609-9. Cited 2 times on pages 16 and 62.

BOOCH, G. Architectural Organizational Patterns. *IEEE Software*, v. 25, n. 3, p. 18–19, May 2008. ISSN 0740-7459. Cited on page 44.

BROOKS, F. P. No silver bullet: Essence and accidents of software engineering. *IEEE Comput.*, p. 10–19, Apr. 1987. Cited on page 15.

CAPILLA, R. et al. 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, v. 116, p. 191–205, Jun. 2016. ISSN 01641212. Available at: <https://linkinghub.elsevier.com/retrieve/pii/S0164121215002034>. Cited on page 17.

CERVANTES, H.; KAZMAN, R. *Designing software architectures: a practical approach*. Boston, MA: Addison-Wesley Educational, 2016. Cited on page 18.

CLEMENTS, P. et al. *Documenting Software Architectures: Views and Beyond*. 2. ed. Boston, MA, USA: Addison-Wesley Educational, 2010. Cited 5 times on pages 15, 16, 17, 20, and 62.

COSTA, J. E. C.; SOARES, M. S. 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*. [S.l.: s.n.], 2023. p. 412–418. Cited 2 times on pages 14 and 71.

ERNST, N. A.; ROBILLARD, M. P. A study of documentation for software architecture. *Empirical software engineering : an international journal*, Springer US, New York, v. 28, n. 5, p. 122, 2023. ISSN 1382-3256. Cited 4 times on pages 16, 17, 18, and 72.

FAIRBANKS, G. *Just enough software architecture*. [S.l.]: Marshall & Brainerd, 2010. Cited 3 times on pages 14, 15, and 16.

GALSTER, M.; WEYNS, D. Empirical research in software architecture — Perceptions of the community. *Journal of Systems and Software*, v. 202, p. 111684, Aug. 2023. ISSN 01641212. Cited on page 34.

GARLAN, D.; SHAW, M. AN INTRODUCTION TO SOFTWARE ARCHITECTURE. In: *Series on Software Engineering and Knowledge Engineering*. [S.l.]: WORLD SCIENTIFIC, 1993. v. 2, p. 1–39. ISBN 9789810215941 9789812798039. Cited 3 times on pages 13, 16, and 34.

GIL, A. C. *Como elaborar projetos de pesquisa*. [S.l.]: Atlas São Paulo, 2002. v. 4. ISBN 9788522431694. Cited on page 19.

HASSELBRING, W. Software Architecture: Past, Present, Future. In: GRUHN, V.; STRIEMER, R. (Ed.). *The Essence of Software Engineering*. Cham: Springer International Publishing, 2018. p. 169–184. ISBN 9783319738963 9783319738970. Cited 2 times on pages 13 and 44.

HEVNER, A. R. et al. Design Science in Information Systems Research. *MIS quarterly*, Management Information Systems Research Center, University of Minnesota, v. 28, n. 1, p. 75–105, 2004. ISSN 0276-7783. Cited on page 20.

HOFMEISTER, C. et al. A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, v. 80, n. 1, p. 106–126, Jan. 2007. ISSN 01641212. Available at: <https://linkinghub.elsevier.com/retrieve/pii/S0164121206001634>. Cited on page 18.

IEEE. *ISO/IEC/IEEE 15288:2015(E) – Systems and Software Engineering — System life cycle processes*. [S.l.], 2015. Cited on page 35.

IEEE. *ISO/IEC/IEEE 12207:2017(E) – Software, systems and enterprise – Architecture description*. [S.l.], 2017. 1–74 p. (ISO/IEC/IEEE International Standard – Software, systems and enterprise). doi: 10.1109/IEEESTD.2022.9938446. Cited on page 17.

IEEE. *ISO/IEC/IEEE 42020:2019(E) – Software, systems and enterprise – Architecture processes*. [S.l.], 2019. 1–126 p. (ISO/IEC/IEEE International Standard – Software, systems and enterprise – Architecture processes). doi: 10.1109/IEEESTD.2019.8767004. Cited 18 times on pages 14, 15, 23, 24, 26, 27, 28, 29, 33, 36, 39, 40, 41, 42, 44, 45, 47, and 60.

IEEE. *ISO/IEC/IEEE 42010:2022(E) – Software, systems and enterprise – Architecture description*. [S.l.], 2022. Cited 7 times on pages 33, 36, 38, 39, 44, 59, and 62.

IVANOV, D. M. Software Architecture Documentation - Guidelines and Tools: Studying guidelines and tools for documenting software architecture effectively to facilitate communication and decision-making. *Journal of Artificial Intelligence Research and Applications*, v. 4, n. 1, p. 82–92, May 2024. Available at: <https://aimlstudies.co.uk/index.php/jaira/article/view/17>. Cited 2 times on pages 58 and 72.

JÚNIOR, A. A. C.; MISRA, S.; SOARES, M. S. ArchCaMO — A Maturity Model for Software Architecture Description Based on ISO/IEC/IEEE 42010:2011. In: MISRA, S. et al. (Ed.). *Computational Science and Its Applications – ICCSA 2019*. Cham: Springer International Publishing, 2019. v. 11623, p. 31–42. ISBN 9783030243074 9783030243081. Cited on page 33.

JUNIOR, A. A. da C.; MISRA, S.; SOARES, M. S. A Systematic Mapping Study on Software Architectures Description Based on ISO/IEC/IEEE 42010:2011. In: MISRA, S. et al. (Ed.). *Computational Science and Its Applications - ICCSA 2019 - 19th International Conference, Saint Petersburg, Russia, July 1-4, 2019, Proceedings, Part V*. [S.l.]: Springer, 2019. (Lecture Notes in Computer Science, v. 11623), p. 17–30. Cited 3 times on pages 33, 39, and 45.

KALSI, H. S. *Importance of Documentation in Software Development and Role of Different Stakeholders*. 2024. Https://medium.com/i-am-a-dummy-enlighten-me/importance-of-documentation-in-software-development-and-role-of-different-stakeholders-799db43ef55d. Acessed: August 15, 2024. Cited on page 17.

KNEUPER, R. Foundations. In: *Software Processes and Life Cycle Models*. Cham: Springer International Publishing, 2018. p. 1–39. Cited 3 times on pages 17, 18, and 49.

KOWALEWSKI, L. *An overview of software architecture documentation*. 2020. Available at: <https://leonkowa.medium.com/an-overview-of-software-architecture-documentation-c9c1a064ba6e>. Cited on page 44.

KRUCHTEN, P. B. The 4+1 View Model of architecture. *IEEE software*, IEEE, v. 12, n. 6, p. 42–50, 1995. ISSN 0740-7459. Cited 3 times on pages 17, 47, and 62.

KUMAR, A.; LOKKU, D. S.; NATARAJAN, S. Architecture Literacy. In: WILEY ONLINE LIBRARY. *INCOSE International Symposium*. [S.l.], 2021. v. 31, n. 1, p. 1319–1333. Cited 2 times on pages 13 and 14.

KUMAR, A.; LOKKU, D. S.; NATARAJAN, S. Architecture Literacy. *INCOSE International Symposium*, v. 31, n. 1, p. 1319–1333, Jul. 2021. ISSN 2334-5837, 2334-5837. Cited 2 times on pages 44 and 45.

LAND, R. A brief survey of software architecture. *Mälardalen Real-Time Research Center (MRTC) Report*, Citeseer, 2002. Cited on page 14.

MAIER, M.; EMERY, D.; HILLIARD, R. Software architecture: introducing IEEE Standard 1471. *Computer*, v. 34, n. 4, p. 107–109, Apr. 2001. ISSN 00189162. Cited 3 times on pages 34, 35, and 38.

MARTIN, J. N. Architecture Definition – A New Process in the ISO International Systems Engineering Standard. *INCOSE International Symposium*, v. 25, n. 1, p. 463–472, Oct. 2015. ISSN 2334-5837, 2334-5837. Cited 3 times on pages 14, 34, and 35.

MARTIN, J. N. Overview of an emerging standard on architecture processes — ISO/IEC/IEEE 42020. In: *2018 Annual IEEE International Systems Conference (SysCon)*. Vancouver, BC: IEEE, 2018. p. 1–8. ISBN 9781538636640. Cited 8 times on pages 24, 33, 36, 39, 41, 42, 44, and 45.

MARTIN, J. N. Aspect-Oriented Architecting Using Architecture Frameworks. *INCOSE International Symposium*, v. 31, n. 1, p. 210–226, Jul. 2021. ISSN 2334-5837, 2334-5837. Cited 4 times on pages 29, 39, 40, and 44.

MARTIN, J. N. Overview of the Revised Standard on Architecture Description – ISO/IEC 42010. *INCOSE International Symposium*, v. 31, n. 1, p. 1363–1376, Jul. 2021. ISSN 2334-5837, 2334-5837. Cited 4 times on pages 33, 38, 39, and 45.

MARTIN, J. N.; O'NEIL, D. P. Enterprise architecture guide for the unified architecture framework (UAF). *INCOSE Int. Symp.*, Wiley, v. 31, n. 1, p. 242–263, Jul. 2021. Cited on page 18.

OZKAYA, M. Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? *Information and software technology*, Elsevier B.V, v. 95, p. 15–33, 2018. ISSN 0950-5849. Cited on page 13.

RIBEIRO, Q. A. D. S.; RIBEIRO, F. G. C.; SOARES, M. S. A Technique to Architect Real-time Embedded Systems with SysML and UML through Multiple Views. In: HAMMOUDI, S. et al. (Ed.). *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 2, Porto, Portugal, April 26-29, 2017*. [S.l.]: SciTePress, 2017. p. 287–294. Cited on page 14.

ROCHA, F. G.; MISRA, S.; SOARES, M. S. Guidelines for Future Agile Methodologies and Architecture Reconciliation for Software-Intensive Systems. *Electronics*, v. 12, n. 7, p. 1582, Mar. 2023. ISSN 2079-9292. Cited 2 times on pages 34 and 35.

ROZANSKI, N.; WOODS, E. *Software systems architecture: working with stakeholders using viewpoints and perspectives*. [S.l.]: Addison-Wesley, 2012. Cited 5 times on pages 17, 51, 61, 62, and 63.

SANTOS, V.; SOARES, M. S. Design and Evaluation using Technology Acceptance Model of an Architecture Conceptualization Framework System based on the ISO/IEC/IEEE 42020. *JUCS - Journal of Universal Computer Science*, v. 29, n. 12, p. 1510–1534, Dec. 2023. ISSN 0948-6968, 0948-695X. Available at: <https://lib.jucs.org/article/104938/>. Cited on page 58.

SANTOS, V. M.; MISRA, S.; SOARES, M. S. Architecture Conceptualization for Health Information Systems Using ISO/IEC/IEEE 42020. In: GERVASI, O. et al. (Ed.). *Computational Science and Its Applications – ICCSA 2020*. Cham: Springer International Publishing, 2020. v. 12254, p. 398–411. ISBN 9783030588168 9783030588175. Cited 5 times on pages 23, 29, 34, 39, and 43.

SIEVI-KORTE, O.; BEECHAM, S.; RICHARDSON, I. Challenges and recommended practices for software architecting in global software development. *Information and Software Technology*, v. 106, p. 234–253, Feb. 2019. ISSN 09505849. Cited on page 33.

SOARES, M. S. *Architecture-Driven Integration of Modeling Languages for the Design of Software-Intensive Systems*. Thesis (Doctor in Philosophy) — Delft University of Technology, 2010. Cited on page 35.

SOARES, M. S. Arquitetura de Software com a norma ISO/IEC/IEEE 42010. *Engenharia de software magazine*, p. 12–16, 2017. Cited on page 14.

SOARES, M. S.; VRANCKEN, J.; WANG, Y. Architecture-Based Development of Road Traffic Management Systems. In: *2010 International Conference on Networking, Sensing and Control (ICNSC)*. [S.l.: s.n.], 2010. p. 26–31. Cited on page 14.

TIAN, F.; LIANG, P.; BABAR, M. A. Relationships between software architecture and source code in practice: An exploratory survey and interview. *Information and Software Technology*, v. 141, p. 106705, Jan. 2022. ISSN 09505849. Available at: <https://linkinghub.elsevier.com/retrieve/pii/S0950584921001580>. Cited on page 17.

WOHLRAB, R. et al. Improving the Consistency and Usefulness of Architecture Descriptions: Guidelines for Architects. In: *2019 IEEE International Conference on Software Architecture (ICSA)*. Hamburg, Germany: IEEE, 2019. p. 151–160. ISBN 9781728105284. Cited 2 times on pages 33 and 41.

Appendix

APPENDIX A – Tasks and Subtasks of the Step 1

Step	Task	Why and How
1.1	Identifying the intended stakeholders of the architecture description	**Concept:** Understanding the stakeholders who will utilize the architecture description is crucial for tailoring the elaboration effort to meet their specific needs; **Method:** Stakeholder interviews and Surveys;
1.2	Defining purpose, scope, objectives, approaches, techniques, methods, and tools for architecture elaboration.	**Concept:** Clearly outlining the goals and boundaries of the elaboration effort ensures that the resulting architecture description aligns with the organization's objectives; Functional and quality (non-functional) requirements can be defined within scope; **Method:** Stakeholder feedback; Documentation review; It may include reviewing industry best practices, standards, and established methodologies for architecture development;
1.3	Establishing metrics for the architecture elaboration effort.	**Concept:** Clearly outlining the goals and boundaries of the elaboration effort ensures that the resulting architecture description aligns with the organization's objectives; **Method:** Stakeholder feedback; Documentation review; It may include reviewing industry best practices, standards, and established methodologies for architecture development;
1.4	Developing an architecture elaboration schedule and defining associated milestones.	**Concept:** Creating a timeline and milestones helps in managing and tracking the progress of the architecture elaboration effort; **Method:** Include start and end dates for each activity, providing a clear roadmap for the entire effort. Moreover, due consideration should be given to potential risks and uncertainties, thereby facilitating the formulation of contingency strategies and permitting adjustments to the schedule in the event of unforeseen challenges during the process of architectural elaboration;

1.5	Develop and Secure Support for the Architecture Elaboration Plan.	**Concept:** Involves creating a comprehensive document that outlines the details and strategies for the architecture elaboration effort; **Method:** Gather essential data and information requisite for the architectural elaboration initiative. Acquire requisite approvals, resources, and funding to execute the proposed plan. After gathering essential data and information requisite for the architectural elaboration initiative described in steps 1-4, compose a comprehensive architecture elaboration plan to acquire the necessary approvals, resources, and funding for executing the proposed plan. Ensure that personnel possess the requisite and pertinent access to this artefact.

Table 6 – Tasks of the step 1 – Preparing the efforts

Step	Task
1.1	Identifying the intended users of the architecture description
1.1.1	List all potential user groups, including architects, project managers, analysts, and stakeholders.
1.1.2	Conduct stakeholder analysis to understand their specific information needs and expectations.
1.1.3	Define a clear set of questions that the architecture should answer, e.g., “How will the system handle scalability?”
1.1.4	Review previous architecture documentation and identify gaps in addressing these questions.
1.1.5	Compile a list of concerns and objectives from stakeholders, prioritizing importance and relevance.
1.2	Defining purpose, objectives, approaches, techniques, methods, and tools for architecture elaboration.
1.2.1	Clearly define the purpose of the architecture elaboration, e.g., “To provide a detailed architectural overview for project planning.”
1.2.2	Set specific objectives, e.g., “To identify system bottlenecks.”
1.2.3	Determine the methodology or approach for architecture elaboration (e.g., TOGAF, Zachman Framework)
1.2.4	Align the approach with organizational architecture governance and management directives.

1.2.5	Assess the availability of architecture methods, techniques and tools aligned with the organization's needs; if absent, choose or define them accordingly. This process entails reviewing best practices and industry/academia standards, while identifying resources for modelling, data analysis, and visualization. Generally, the objective is to establish methodologies and tools conducive to creating and documenting architectural models and visualizations.
1.3	Establishing metrics for the architecture elaboration effort.
1.3.1	Defining indicators for tracking progress, identifying areas for improvement, and quantitatively or qualitatively measuring the developed architecture description.
1.3.2	Establish metrics that include measurement frequency, units, and ideal target values (if applicable).
1.4	Developing an architecture elaboration schedule and defining associated milestones.
1.4.1	Creating a schedule that outlines the sequence of activities, tasks, and events involved in the architecture elaboration process.
1.4.2	Organizing the activities in a logical sequence, ensuring that dependencies between tasks are identified and accounted for in the schedule.
1.4.3	Identifying specific milestones that represent significant points of progress or achievement within the architecture elaboration effort.
1.5	Develop and Secure Support for the Architecture Elaboration Plan.
1.5.1	Consolidate the information gathered into a detailed plan.
1.5.2	Ensure data is organized and accessible to the project team.
1.5.3	Define access levels and permissions for team members.
1.5.4	Provide training sessions or resources.

Table 7 – Subtasks of the step 1 – Preparing the efforts

APPENDIX B – Tasks and Subtasks of the Step 2

Step	Task	Why and How
2.1	Identify, adapt or develop viewpoints and model kinds based on stakeholder concerns.	**Concept:** The process entails understanding stakeholder needs and crafting specialized perspectives to fulfill architectural requirements. It involves a thorough analysis of stakeholders and users to tailor architecture descriptions, models, and data accordingly. This ensures that the resulting information effectively meets their specific needs; **Method:** Identify and adapt existing viewpoints, possibly from prior architecture efforts or frameworks, to suit the current context. Develop new viewpoints if existing ones are insufficient or new perspectives are needed. This includes creating modelling methods and templates tailored to specific concerns like operational, security, services, or capability viewpoints, meeting various stakeholder needs.
2.2	Identify, adapt or develop architecture frameworks, modelling templates and metamodels.	**Concept:** The identification, adaptation, or development of architecture viewpoints and model kinds, should align with established architecture frameworks, ensuring that the viewpoints conform to the conventions and standards set forth by the framework being utilized. **Method:** Initiates with evaluating frameworks (e.g., TOGAF, DODAF, Zachman Framework), understanding project needs, and deciding on customization. A choice between adapting existing frameworks or creating new ones follows. Designing modelling templates and defining a metamodel that adheres to the selected framework ensures industry standards compliance. Documenting and communicating the chosen approach, along with training sessions, enhance personnel comprehension.

2.3	Prepare for developing view(s).	**Concept:** Define the purpose and scope of each model and view to be developed or adapted; **Method:** Begin with a contextual analysis and stakeholder identification generated for Step 1, followed by clearly defining overall objectives. The selection of appropriate models and views is based on these objectives, and their purpose is articulated, specifying the information they aim to convey. The scope of each representation is then defined to ensure a focused and manageable coverage. Alignment with industry standards, validation through reviews, and comprehensive documentation contribute to effective communication.

Table 8 – Tasks of the step 2 – Designing the viewpoints

Step	Task
2.1	Identify, adapt or develop viewpoints and model kinds based on stakeholder concerns.
2.1.1	Review stakeholder concerns: Begin by identifying and reviewing the concerns of relevant stakeholders. This involves engaging with stakeholders and gathering their specific concerns regarding the architecture.
2.1.2	Analyze existing viewpoints and model kinds: Investigate if any existing architecture viewpoints address similar stakeholder concerns. Determine whether these can be adapted or reused.
2.1.3	Adapt or develop new viewpoints and model kinds if needed: If no existing viewpoints are suitable, create new ones based on the identified stakeholder concerns. Design these viewpoints to effectively address the specific concerns.
2.1.4	Identify the target audience and assess their requirements: Determine who the intended users of the architecture elaboration information will be. This can include architects, stakeholders, analysts, project managers, and other relevant parties. Understand the specific needs and requirements of the identified users. What kind of information or views do they require for decision-making or analysis?
2.1.5	Document the reasoning: Record the rationale behind of the viewpoints and model kinds.
2.2	Identify, adapt or develop architecture frameworks, modelling templates and metamodels
2.2.1	Determine architecture description framework(s): Examine if there are any frameworks already in use or previously identified during the Architecture Conceptualization process that are suitable for developing the required architecture views and models. If needed, establish potential frameworks for guidance for model and view development.
2.2.2	Select or customize frameworks: If existing frameworks are not appropriate, select or customize a framework that aligns with the architecture objectives and stakeholder needs.

2.2.3	Select, modify or develop supporting modelling methods and tools, modifying metamodels which ensure that the chosen architecture frameworks can accommodate the required modelling elements and relationships.
2.2.4	Choose appropriate modelling languages, adapt existing models, or create new ones based on stakeholder needs.
2.2.5	Define the rationale: If defined architecture framework(s), clearly document how the framework was identified or established, including how it addresses stakeholder concerns and fulfils the architecture objectives.
2.3	Prepare for developing view(s).
2.3.1	Set the objectives: Define the purpose of each model and view that will be developed during the architecture elaboration process. This should align with the stakeholder concerns and architecture objectives.
2.3.2	Specify scope: Clearly outline the scope of each model and view, including the extent to which they will address architectural concepts, properties, characteristics, behaviours, functions, features, or constraints.
2.3.3	Assess existing tools: Review any existing modelling methods and tools within the organization. Determine if they can be employed for developing the architecture views.
2.3.4	Acquire or develop tools: If existing tools are insufficient, select, acquire, or develop new modelling methods and tools that align with the chosen viewpoints and model kinds.

Table 9 – Subtasks of the step 2 – Designing the viewpoints

APPENDIX C – Tasks and Subtasks of the Step 3

Step	Task	Why and How
3.1	Define architectural context and boundaries for each viewpoint.	**Concept:** Defining context and boundaries elucidates the scope and purpose of a viewpoint, addressing stakeholders' concerns effectively. **Method:** Commence with stakeholder analysis, selecting relevant viewpoints. Define clear objectives for each viewpoint, delineating focused boundaries. Identify key components and interactions, aligning with the overarching architecture while considering external factors. Thorough documentation, stakeholder validation, and regular review mechanisms ensure current documentation status.
3.2	Identify entities and relationships for modelling.	**Concept:** Identification of entities and relationships ensures alignment with stakeholder concerns and architecture objectives. **Method:** Initiate with stakeholder engagement and review of project requirements. Define modelling scope, prioritize key entities, and explore relationships considering dependencies and interactions. Align with architecture viewpoints, utilize appropriate notations, and document comprehensively. Stakeholder reviews validate model accuracy, with continuous refinement adapting to evolving project needs.
3.3	Allocate concepts, properties, and characteristics to modelled Entities.	**Concept:** Allocating concepts ensures that the modelled entities and relationships are accurately represented based on relevant characteristics. **Method:** Attribute analysis, requirements allocation, and domain expertise.
3.4	Compose views to address stakeholder concerns and architecture objectives.	**Concept:** Composing views translates complex models into understandable representations, addressing stakeholder concerns and requirements. **Method:** Use defined viewpoints, arrange model elements, and ensure clarity and relevance.
3.5	Harmonize architecture views.	**Concept:** Harmonizing ensures consistency and coherence among different views, improving overall clarity and understanding. **Method:** Conduct views reviews, identify inconsistencies and align representations.

Table 10 – Tasks of the step 3 – Developing the views

Step	Task
3.1	Define Architectural Context and Boundaries for Each Viewpoint.
3.1.1	Determine interfaces and interactions: Identify the architectural context and establish boundaries for each selected viewpoint. Define how the architecture interacts with external entities and interfaces.
3.1.2	Establish viewpoint purpose and scope: Clearly articulate the purpose and scope of each viewpoint. This should include the specific concerns they aim to address and the aspects they will encompass.
3.2	Identify Entities and Relationships for Modeling.
3.2.1	Identify modelling entities: Determine the types of entities within the architecture that need to be modelled to address key stakeholder concerns and meet architecture objectives.
3.2.2	Specify relationships: Define the relationships between these identified entities. These relationships should capture the interactions and dependencies critical to the architecture.
3.3	Allocate concepts, properties, and characteristics to modelled entities.
3.3.1	Allocate relevant architectural concepts, properties, characteristics, behaviours, functions, features, or constraints to the entities being modelled. Ensure that these allocations align with the architecture objectives.
3.4	Compose Views to Address Stakeholder Concerns and Architecture Objectives.
3.4.1	Use the selected models to generate architecture views that represent how the architecture addresses stakeholder concerns, principles, objectives, and requirements. Ensure that each view aligns with its associated viewpoint.
3.5	Harmonize Architecture Models and Views.
3.5.1	Ensure consistency: Review all architecture models and views to ensure they are consistent with each other and that there are no inconsistencies or contradictions in the representations.

Table 11 – Subtasks of the step 3 – Developing the views

APPENDIX D – Tasks and Subtasks of the Step 4

Step	Task	Why and How
4.1	Evaluating architecture views and models in achitectural elaboration	**Concept:** This phase involves assessing every architecture view and model concerning the predefined objectives, purposes, and inquiries of the architectural elaboration process. **Method:** Conduct a systematic review and evaluation validating the developed architecture against the initial conceptualized intent, actively identifying and resolving any discrepancies or misalignments encountered. This thorough evaluation ensures coherence between the envisioned architecture and its actual implementation, enhancing the effectiveness and fidelity of the architectural endeavour.
4.2	Evaluation and Verification of Architecture Descriptions and Work Products	**Concept:** Ensures that the elaborated architecture aligns with the original vision, identifying and resolving any inconsistencies. **Method:** Conduct a comparative analysis between the elaborated architecture and the initially conceptualized architecture.
4.3	Assess the suitability of the architecture description.	**Concept:** Ensures that the architecture description effectively communicates the necessary information. **Method:** Evaluate the architecture description against defined criteria for completeness, clarity, and relevance.
4.4	Verify the consistency of architecture work products.	**Concept:** Ensures alignment and coherence across various documentation and depictions. **Method:** Cross-reference architecture work products with related documentation and depictions.
4.5	Update the architecture description to address identified redundancies, gaps, and shortfalls.	**Concept:** Continuous improvement to eliminate redundancies and enhance completeness. **Method:** Interactively review and revise the architecture description based on assessment findings.

4.6	Identify views and models that can be generalized for reuse by other projects.	**Concept:** Promotes the reuse of successful architectural elements in different contexts. **Method:** Evaluate architecture views and models for their potential to be applied in other projects.
4.7	Place updated architecture description in the architecture repository and under change control, when appropriate.	**Concept:** Centralized storage for easy access and version control. Ensures that changes are controlled and documented for traceability. **Method:** Ensure the revised architecture description is stored in a designated repository. Apply change control procedures to manage modifications to the architecture description.

Table 12 – Tasks of step 4 – Assessing the document (Architecture description)

Step	Task
4.1	Assess each architecture view and model.
4.1.1	Develop a checklist of objectives and questions for each architecture view.
4.1.2	Evaluate each view against its specific checklist.
4.1.3	Document findings and recommendations.
4.2	Validate elaborated architecture.
4.2.1	Define criteria for comparing the elaborated architecture with the conceptualized architecture.
4.2.2	Identify any discrepancies or deviations.
4.2.3	Initiate corrective actions to resolve identified issues.
4.3	Assess the suitability of the architecture description.
4.3.1	Define criteria for assessing the architecture description.
4.3.2	Conduct a detailed review of the architecture description.
4.3.3	Provide feedback and recommendations for improvement.
4.4	Verify the consistency of architecture work products.
4.4.1	Identify relevant external documentation and depictions.
4.4.2	Verify consistency between architecture work products and external references.
4.4.3	Document any inconsistencies and propose corrective actions.
4.5	Update the architecture description to address identified redundancies, gaps, and shortfalls.
4.5.1	Analyze assessment findings related to redundancies, gaps, and shortfalls.
4.5.2	Revise the architecture description accordingly.
4.5.3	Document version changes and updates.

4.6	Identify views and models that can be generalized for reuse by other projects.
4.6.1	Define criteria for assessing the generalizability of views and models.
4.6.2	Identify views and models suitable for reuse.
4.6.3	Document guidelines for reuse.
4.7	Place updated architecture description in the architecture repository and under change control, when appropriate.
4.7.1	Verify the integrity of the updated architecture description.
4.7.2	Upload the revised version to the architecture repository.
4.7.3	Apply change control processes as per the significance assessment.

Table 13 – Subtasks of the step 4 - Assessing the architecture description (Document)

APPENDIX E – Tasks and Subtasks of the Step 5

Step	Task	Why and How
5.1	Identify users of architecture elaboration information.	**Concept:** This task aims to understand and define the audience for the architecture elaboration information, ensuring that the delivered information meets the needs and expectations of various stakeholders. **Method:** Conduct stakeholder analysis, interviews, and surveys to identify and categorize users. Utilize collaboration tools and communication channels to gather insights from potential users.
5.2	Prepare architecture elaboration information and data for use by others.	**Concept:** This task involves organizing and presenting the architecture information in a clear and accessible manner, aligning with the diverse needs of users. **Method:** Develop user-friendly documentation, create tailored presentations, and ensure that the information is structured according to user requirements. Leverage visualization tools to enhance understanding.
5.3	Maintain the architecture elaboration information and data during use.	**Concept:** Ongoing maintenance ensures that the information remains accurate, relevant, and effective as users engage with it. **Method:** Implement version control, establish feedback mechanisms, and continuously update the information based on user interactions. Use change management processes to handle modifications.
5.4	Maintain change control of architecture elaboration-related data items.	**Concept:** Change control ensures that alterations to architecture information are managed systematically, preventing errors and maintaining integrity. **Method:** Enforce a change control process, requiring approval for modifications. Communicate changes to relevant stakeholders promptly.

5.5	Deliver architecture elaboration information to intended users.	**Concept:** Timely delivery ensures that users have access to the most current and relevant architecture information. **Method:** Use established delivery channels, such as direct distribution, repository updates, or formal release processes. Validate information before release.
5.6	Monitor the use of architecture elaboration information.	**Concept:** Monitoring usage provides valuable feedback for continuous improvement and ensures that the architecture meets user expectations. **Method:** Implement tracking mechanisms, collect user feedback, and analyze usage patterns. Regularly review the effectiveness of the architecture information.
5.7	Communicate architecture elaboration information to interested parties.	**Concept:** Communication broadens awareness and understanding of the architecture, fostering collaboration and alignment. **Method:** Use various communication channels, such as newsletters, meetings, or webinars, to reach interested parties. Tailor messages to address specific concerns. **Tool(s):** Communication platforms, presentation tools.
5.8	Incorporate feedback into the architecture descriptions, views, and models.	**Concept:** Integrating feedback ensures that the architecture evolves based on user insights, improving its quality and relevance. **Method:** Establish a feedback loop, prioritize actionable feedback, and update architecture components accordingly. Communicate changes to users.

Table 14 – Tasks of the step 5 - Coordinating Usage

Step	Task
5.1	Identify users of architecture elaboration information.
5.1.1	Classify potential users based on their roles, interests, and influence on the architecture.
5.1.2	Engage in one-on-one or group discussions with key stakeholders to understand their information needs and expectations.
5.1.3	Create and distribute surveys to a broader audience to gather quantitative data on user preferences and requirements.
5.2	Prepare architecture elaboration information and data for use by others.
5.2.1	Organize architecture information into logical sections, ensuring clarity and easy navigation.

5.2.2	Develop presentations that address specific user needs, emphasizing relevant aspects of the architecture.
5.2.3	Enhance understanding through visual aids, such as diagrams, charts, and models.
5.3	Maintain the architecture elaboration information and data during use.
5.3.1	Establish a versioning system to track changes and ensure users access the latest information.
5.3.2	Set up mechanisms for users to provide feedback on the architecture, encouraging continuous improvement.
5.3.3	Regularly review and update the information based on user feedback and changing requirements.
5.4	Maintain change control of architecture elaboration-related data items.
5.4.1	Establish clear procedures for requesting, reviewing, and approving changes to architecture information.
5.4.2	Inform relevant stakeholders promptly about approved changes, highlighting the nature and impact of modifications.
5.5	Deliver architecture elaboration information to intended users.
5.5.1	Select appropriate channels for information delivery, considering the preferences of different user groups.
5.5.2	Ensure the accuracy and completeness of information before distribution.
5.5.3	Adhere to formal release processes if applicable, verifying that the information meets organizational standards.
5.6	Monitor the use of architecture elaboration information.
5.6.1	Integrate analytics tools or tracking systems to monitor user interactions with architecture information.
5.6.2	Encourage ongoing feedback through surveys, user forums, or direct communication.
5.6.3	Regularly review usage patterns to identify popular sections, potential issues, and areas for improvement.
5.7	Communicate architecture elaboration information to interested parties.
5.7.1	Select appropriate channels for reaching different interested parties, considering their preferences.
5.7.2	Develop messages that address the specific concerns and interests of each target audience.
5.7.3	Maintain a regular communication cadence to keep stakeholders informed about architecture developments.
5.8	Incorporate feedback into the architecture descriptions, views, and models.
5.8.1	Create a structured process for receiving, prioritizing, and acting on user feedback.
5.8.2	Identify feedback with the most significant impact on architecture quality and prioritize updates accordingly.

5.8.3	Clearly communicate changes made in response to feedback, ensuring transparency and user awareness.

Table 15 – Subtasks of the step 5 - Coordinating Usage

APPENDIX F – Tasks and Subtasks of the Step 6

Step	Task	Why and How
6.1	Identify related entities and other architectures that relate to architecture elements and the nature of these relationships.	**Concept:** Understanding the ecosystem and interdependencies ensures comprehensive consideration during architecture development. **Method:** Conduct stakeholder interviews, surveys, and analysis to identify entities and architectures relevant to the system.
6.2	Define the interfaces and interactions between the related entities and other architectures with each other and with the architecture being elaborated.	**Concept:** Clearly defined interfaces facilitate seamless communication and integration. **Method:** Organize workshops or collaborative sessions to define interfaces and interactions.
6.3	Identify areas for potential reuse of existing architecture elements and the risks associated with this reuse.	**Concept:** Reuse promotes efficiency, but risks must be identified and mitigated. **Method:** Perform a thorough analysis of existing architecture elements and assess their suitability for reuse.
6.4	Partition, align and allocate requirements to architecture elements and these related entities.	**Concept:** Requirements alignment ensures that each architecture element contributes to the fulfilment of specified needs. **Method:** Use requirement analysis techniques to allocate and align requirements.
6.5	Map related entities and other architectures to relevant architecture concepts, properties and other attributes.	**Concept:** Mapping enhances clarity on the impact and influence of related entities. **Method:** Utilize mapping techniques to associate entities with relevant architectural concepts and properties.

6.6	Formulate principles and precepts expected to be used during the execution of the life cycle processes for the architecture entity.	**Concept:** Principles guide decision-making and actions throughout the life cycle. **Method:** Facilitate workshops and discussions to derive guiding principles and precepts.
6.7	Formulate principles and precepts for the design and evolution of the architecture entity.	**Concept:** Design and evolution principles shape the long-term viability of the architecture. **Method:** Engage stakeholders in envisioning the design and evolution principles.

Table 16 – Tasks of step 6 – Connecting other architectures and relevant entities

Step	Task
6.1	Identify related entities and other architectures that relate to architecture elements and the nature of these relationships.
6.1.1	Conduct stakeholder interviews to gather insights on related entities.
6.1.2	Analyze existing documentation to identify previously established relationships.
6.1.3	Use visual mapping tools to represent identified relationships.
6.2	Define the interfaces and interactions between the related entities and other architectures with each other and with the architecture being elaborated.
6.2.1	Facilitate workshops to gather input on interfaces and interactions.
6.2.2	Document interface specifications and interaction protocols.
6.2.3	Validate interface definitions with relevant stakeholders.
6.3	Identify areas for potential reuse of existing architecture elements and the risks associated with this reuse.
6.3.1	Evaluate existing architecture elements for reuse potential.
6.3.2	Identify and assess risks associated with reuse.
6.3.3	Develop strategies to mitigate identified risks.
6.4	Partition, align and allocate requirements to architecture elements and these related entities.
6.4.1	Analyze and understand system requirements.
6.4.2	Allocate requirements to specific architecture elements.
6.4.3	Ensure alignment with related entities' requirements.
6.5	Map related entities and other architectures to relevant architecture concepts, properties and other attributes.
6.5.1	Establish a mapping framework for entities and architecture concepts.

6.5.2	Map each related entity to relevant architectural attributes.
6.5.3	Review and validate the mappings with stakeholders.
6.6	Formulate principles and precepts expected to be used during the execution of the life cycle processes for the architecture entity.
6.6.1	Gather input from stakeholders on desired principles.
6.6.2	Formulate clear and concise principles based on collected input.
6.6.3	Document and communicate principles to relevant stakeholders.
6.7	Formulate principles and precepts for the design and evolution of the architecture entity.
6.7.1	Conduct collaborative sessions to define design and evolution principles.
6.7.2	Draft and refine principles based on stakeholder feedback.
6.7.3	Establish a mechanism for incorporating principles into the architecture life cycle.

Table 17 – Subtasks of step 6 – Connecting other architectures and relevant entities

APPENDIX G – Tasks and Subtasks of the Step 7

Table 18 – Tasks of step 7 - Supervising the architecture elaboration

Step	Task	Why and How
7.1	Report architecture elaboration activity plans and status.	**Concept:** Reporting plans and status provides transparency and accountability, ensuring all stakeholders are informed about the progress and upcoming activities. **Method:** Use project management software to create and manage the activity plan. Conduct regular status meetings to discuss progress, challenges, and next steps. Create dashboards using business intelligence tools or spreadsheet software. Regularly send email updates and notifications to team members.
7.2	**Description:** Monitor and assess whether architecture governance directives and guidance are being followed.	**Concept:** Ensuring adherence to governance directives maintains consistency, compliance, and alignment with organizational standards. **Method:** Analyze governance policies and assess their relevance. Implement a compliance tracking system. Plan and execute compliance audits. Schedule governance meetings.
7.3	Monitor and assess whether architecture management directives and guidance are being followed.	**Concept:** Consistent application of management directives enhances efficiency, coordination, and overall effectiveness. **Method:** Analyze management policies for alignment. Implement a compliance tracking system.
7.4	Monitor and assess metrics for the architecture elaboration effort.	**Concept:** Metrics provide quantitative insights into the progress, quality, and efficiency of the architecture elaboration process. **Method:** Use data collection techniques to gather metrics. Data analysis and visualization techniques. Distribute metric reports during project meetings.

7.5	Identify and assess risks and opportunities associated with the architecture elaboration effort.	**Concept:** Identifying risks and opportunities enables proactive management, mitigating potential issues and capitalizing on favourable conditions. **Method:** Use techniques like risk workshops or SWOT analysis. Utilize methods like brainstorming sessions. Create and update logs for tracking. Facilitate workshops involving relevant stakeholders.
7.6	Implement risk mitigation efforts for risks deemed sufficiently critical to warrant such action.	**Method:** Use risk mitigation planning techniques. Implement risk response actions. Assess the effectiveness of mitigation measures. **Tool(s):** Risk mitigation templates or planning software. Project management and tracking tools. Impact analysis tools or project management software.
7.7	Implement opportunity pursuit efforts for opportunities deemed sufficiently worthy to warrant such action.	**Method:** Utilize opportunity pursuit planning techniques. Implement opportunity actions. Assess the effectiveness of opportunity actions.
7.8	Maintain traceability of architecture elaboration results to the source material used during the elaboration effort.	**Method:** Define a traceability framework and process. Create and update traceability matrices. Periodically review architecture documentation.
7.9	Ensure that the architecture description is maintained.	**Method:** Define version control guidelines. Define documentation management procedures.
7.10	Ensure that the architecture description is under proper change control.	**Method:** Define change request procedures. Define configuration management procedures.
7.11	Provide developed or modified viewpoints to the Architecture Enablement process for possible use as an organizational standard.	**Method:** Evaluate the effectiveness and relevance of viewpoints. Submit viewpoints to the standardization process.

7.12	Provide developed or modified viewpoints to the Architecture Enablement process for possible use as an organizational standard.	**Method:** Implement usage monitoring and reporting. Conduct training sessions and workshops.
7.13	Ensure that other processes are properly using architecture elaboration products.	**Method:** Review process integration and compliance. Collect feedback and suggestions from users.

Table 19 – Tasks of step 7 – Supervising the elaboration

Step	Task
7.1	Report architecture elaboration activity plans and status.
7.1.1	Regularly update and maintain a record of the architecture elaboration activity plan, including milestones, schedules, and resources required.
7.1.2	Hold periodic status meetings to review the progress of the architecture elaboration activities.
7.1.3	Develop and maintain dashboards or visual reports to provide a quick overview of activity status.
7.1.3	Use email communication for quick updates and notifications among the project team.
7.2	Monitor and assess whether architecture governance directives and guidance are being followed
7.2.1	Review governance policies to ensure alignment with architecture objectives.
7.2.2	Track compliance with governance policies by monitoring activities.
7.2.3	Conduct periodic audits to validate compliance with governance policies.
7.2.3	Hold governance meetings to discuss alignment and receive feedback.
7.3	Monitor and assess whether architecture management directives and guidance are being followed.
7.3.1	Review management policies to ensure they are aligned with the architecture objectives.
7.3.2	Track compliance with management policies through monitoring activities.
7.3.3	Conduct periodic audits to validate compliance with management policies.
7.3.3	Schedule management meetings to discuss alignment and gather feedback.
7.4	Monitor and assess metrics for the architecture elaboration effort.
7.4.1	Collect relevant metrics related to the architecture elaboration effort.
7.4.2	Analyze collected data to generate reports and insights.
7.4.3	Share metric reports and findings with the project team.

7.5	Identify and assess risks and opportunities associated with the architecture elaboration effort.
7.5.1	Utilize risk assessment methods to identify potential risks.
7.5.2	Apply opportunity assessment methods to identify potential opportunities.
7.5.3	Maintain risk and opportunity logs to document identified risks and opportunities.
7.5.3	Organize risk and opportunity workshops for detailed analysis and mitigation planning.
7.6	Implement risk mitigation efforts for risks deemed sufficiently critical to warrant such action.
7.6.1	Develop detailed risk mitigation plans for identified critical risks.
7.6.2	Execute planned risk response actions to mitigate identified risks.
7.6.3	Analyze the impact of risk mitigation measures and adjust plans as needed.
7.7	Implement opportunity pursuit efforts for opportunities deemed sufficiently worthy to warrant such action.
7.7.1	Create detailed opportunity pursuit plans for identified valuable opportunities.
7.7.2	Execute planned opportunity actions to seize valuable opportunities.
7.7.3	Analyze the impact of opportunity actions and adjust plans as necessary.
7.8	Maintain traceability of architecture elaboration results to the source material used during the elaboration effort.
7.8.1	Establish a traceability management process to link results to source material.
7.8.2	Maintain traceability matrices that detail the relationships between results and source material.
7.8.3	Regularly review documentation to ensure that traceability is maintained.
7.9	Ensure that the architecture description is maintained.
7.9.1	Implement document version control procedures to manage changes.
7.9.2	Manage architecture documentation through a structured documentation management process.
7.10	Ensure that the architecture description is under proper change control.
7.10.1	Implement a change request process for architecture documents.
7.10.2	Manage document changes through a structured configuration management process.
7.11	Provide developed or modified viewpoints to the Architecture Enablement process for possible use as an organizational standard.
7.11.1	Assess the suitability of developed or modified viewpoints for standardization.
7.11.2	Initiate standardization processes for selected viewpoints.
7.12	Provide developed or modified viewpoints to the Architecture Enablement process for possible use as an organizational standard.
7.12.1	Monitor the usage of architecture elaboration products by the project team.
7.12.2	Provide training to team members on the effective use of architecture elaboration products.

7.13	Ensure that other processes are properly using architecture elaboration products.
7.13.1	Assess the integration of architecture elaboration products into other relevant processes.
7.13.2	Gather feedback from other processes to improve the quality and relevance of architecture elaboration products.

Table 20 – Subtasks of step 7 – Supervising the elaboration

Annex

ANNEX A – Architecture Elaboration Plan

ArchCE Project

Background

ArchCE (Architecture Conceptualization and Elaboration) functions as a web application. Users provide information about a specific entity, and ArchCE processes this input to generate various architectural work products, including models that visually represent the system's structure and detailed descriptions that explain its functionalities.

Purpose

In contrast to most tools reviewed by \citeonline{IVANOV:2024}, which vary from basic diagramming tools to sophisticated documentation management systems, each offering distinct features and capabilities, ArchCE integrates some characteristics aligned with the developed process. Consequently, it adheres to the guidelines of ISO/IEC/IEEE 42020 and ISO/IEC/IEEE 42010.

Scope

A first version will be developed using basic elements and without a expressive UI. Begins with the preparation of efforts (step 1), including the Architecture Elaboration Plan as a work product. This step is followed by the design of viewpoints (step 2), which includes the Architecture Viewpoint Report as a work product. The subsequent step is the development of views (step 3), which includes the Architecture Views Report as a work product. Finally, the assessment of the document (step 4) is undertaken, which includes the final work product, the Architecture Description.

Function requirements

Code	Requirement description
FR01	The system should allow the User management of the Entity of Interest
FR02	The system should allow the User management of the Stakeholders
FR03	The system should allow the User to manage the Viewpoints
FR04	The system should allow the User to manage the Views

Objectives

- Demonstrates how it is possible to spend minimal time to understand the extensive tasks related to Clause 10 (Architecture Elaboration) of the ISO/IEC/IEEE 42020 standard
- Serve as a reference for software architecture education and training.

Stakeholders

- Carlos Santana
- Amilton dos Anjos
- Bill Gates
- Bill Gates

Approach

1. Evaluation and Selection:

    - We will evaluate existing architecture frameworks (e.g., TOGAF, Zachman Framework) based on their suitability for the project's complexity and domain.
    - We will assess available architecture elaboration techniques (e.g., workshops, scenario modeling) and identify the most efficient methods for gathering requirements and defining the architecture.
    - We will review pre-defined architecture viewpoints, modeling templates, and view generation methods to ensure they capture the necessary information for stakeholders.
    - If existing options are not ideal, we will consider developing or modifying frameworks, templates, and methods to suit the project's specific needs. This should be done with justification and documented for future reference.

2. Tailored Architecture Definition:

    - The chosen framework will be used as a foundation, but we will have the flexibility to adapt it to the project's specific requirements.
    - We will select appropriate viewpoints (e.g., functional, data) to document the architecture from different perspectives, ensuring all stakeholders have a clear understanding.
    - We will leverage existing modeling templates or develop customized ones to capture the necessary details of the architecture components, their interactions, and data flows.
    - View generation methods will be chosen based on their effectiveness in communicating the architecture to different audiences (e.g., diagrams for technical teams, narrative descriptions for stakeholders).

3. Documentation and Communication:

    - The architecture will be comprehensively documented, including decisions made, justification for choices, and a clear traceability matrix linking requirements to architecture components.
    - We will utilize appropriate tools for documenting the architecture, potentially leveraging existing enterprise architecture tools or collaborative platforms depending on governance guidelines.
    - Effective communication strategies will be employed to present the architecture to stakeholders. This might involve workshops, presentations, and interactive visualizations to ensure understanding and buy-in.

4. Continuous Improvement:

    - We will adopt an iterative approach to architecture elaboration, allowing for refinements based on stakeholder feedback or changing project requirements.
    - The chosen architecture approach will be reviewed and assessed after project completion to identify areas for improvement for future projects.

Schedule

- Week 1-2: Requirements Analysis and Framework Selection (Milestone 1: Requirements Baseline)
    - Activities:
        - Analyze system requirements for functionality, performance, security, and scalability.
        - Identify key architectural drivers based on the requirements analysis.
        - Research and evaluate potential architecture frameworks and technologies that align with project goals, architecture governance, and available resources.
- Week 3-4: High-Level Architecture Definition (Milestone 2: Initial Architecture Definition)
    - Activities:
        - Select the most suitable architecture framework based on the evaluation in Week 1-2.

        - Define the overall system architecture, including layers (e.g., presentation, business logic, data access) and communication protocols.
        - Identify major architectural components and their interactions.
- Week 5-6: Technology Stack Selection and Integration Strategy (Milestone 3: Technology Stack Defined)
    - Activities:
        - Identify specific technologies (e.g., databases, programming languages) that align with the chosen framework and project requirements.
        - Define a strategy for integrating third-party APIs and external systems.
        - Consider infrastructure needs (e.g., cloud deployment) and security implications.
- Week 7-8: Detailed Architecture Design and Refinement (Milestone 4: Detailed Architecture Defined)
    - Activities:
        - Refine the architecture based on selected technologies and identified integration points.
        - Design detailed component blueprints with interfaces, functionalities, and data flows.
        - Consider performance optimization strategies and potential scalability needs.
- Week 9-10: Communication and Documentation (Milestone 5: Architecture Review and Sign-Off)
    - Activities:
        - Document the complete architecture, including decisions made, technologies chosen, and integration points.
        - Present the architecture to stakeholders for review and feedback.
        - Address any concerns and incorporate feedback into the final architecture documentation.

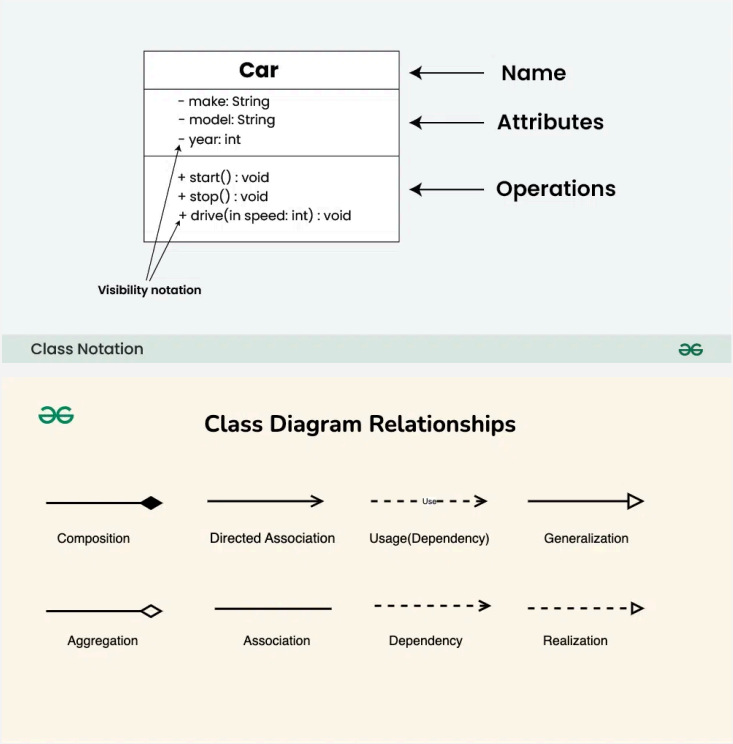
Milestones

- Milestone 1: Requirements Baseline
    - Deliverable: Documented set of system requirements, prioritized and categorized.
- Milestone 2: Initial Architecture Definition
    - Deliverable: High-level architecture diagram with component descriptions, interactions, and chosen framework.
- Milestone 3: Technology Stack Defined
    - Deliverable: Documented technology stack with justifications for selections and an integration strategy document.
- Milestone 4: Detailed Architecture Defined
    - Deliverable: Detailed architecture documentation with component diagrams, specifications, and performance considerations.
- Milestone 5: Architecture Review and Sign-Off
    - Deliverable: Finalized architecture documentation with stakeholder approvals.

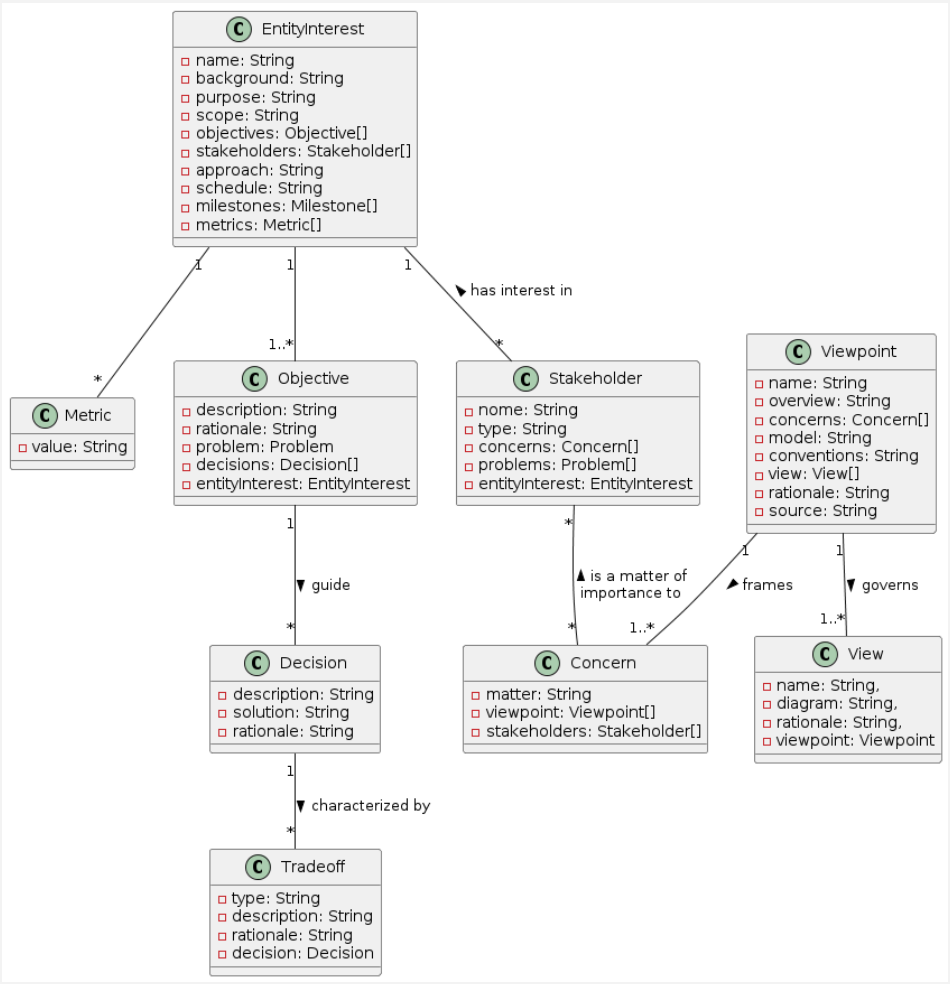
ANNEX B – ArchCE Project — Architecture viewpoints and views

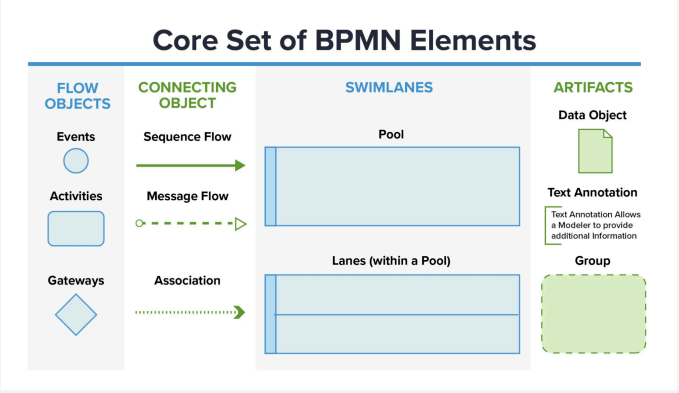
Architecture viewpoints and views

ArchCE Project

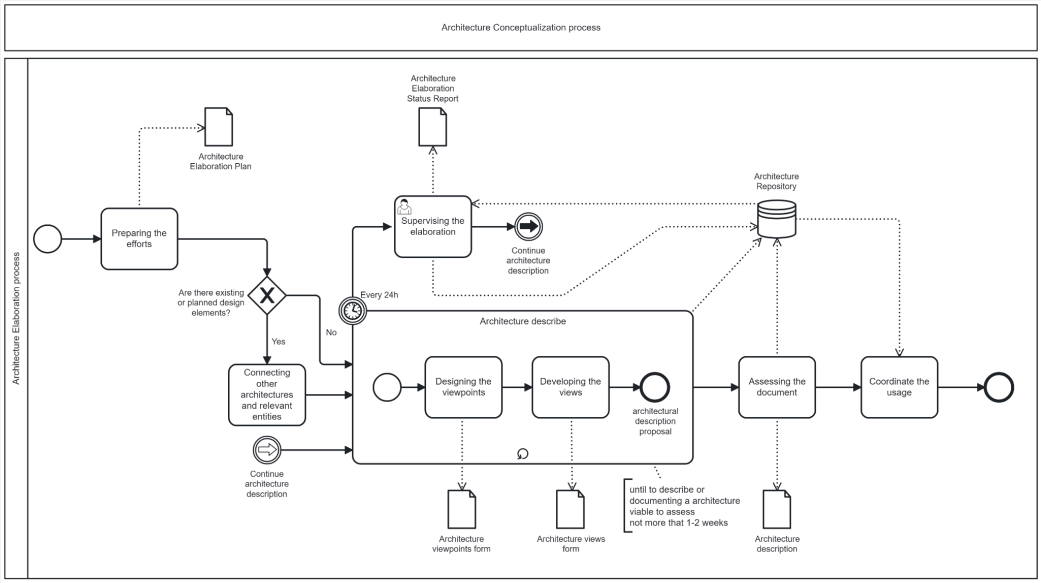
Information viewpoint	
Overview:	Describes the way that the system stores, manipulates, manages, and distributes information
Concerns:	Information Structure and Content
Stakeholder(s):	Bill Gates
Stakeholder perspective(s):	Development
Views:	Context view
Models:	**Static Information Structure Model** The diagram will be illustrated using UML notation with the appropriate use of classes, relationships between classes, stereotypes, and others). 

Information view	
Viewpoint:	Information viewpoint
Concern:	Information Structure and Content



Concurrency viewpoint	
Overview:	Describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled.
Concern(s):	Business state
Stakeholder(s):	Bill Gates
Stakeholder perspective(s):	Development
View(s):	Business Process View
Model(s):	**Business Process Model** A basic process in the business process model would normally contain the following types of entities: 

Business Process view	
Viewpoint:	Concurrency viewpoint
Concern(s):	Business State



Development viewpoint	
Overview:	Describes the architecture that supports the software development process
Concern(s):	Module Organization
Stakeholder(s):	Bill Gates
Stakeholder perspective(s):	Development
View(s):	Development view
Model(s):	**Module Structure Models** The diagram is represented with a UML component diagram, using the package icon to represent a code module and dependency arrows to show intermodule dependencies. **Notation** 
Source:	UML2 Version; Rozanski and Woods (2012)

Development view	
Viewpoint:	Development viewpoint
Concern:	Module Organization