FEDERAL UNIVERSITY OF SERGIPE

CENTER OF EXACT SCIENCES AND TECHNOLOGY

POSTGRADUATE PROGRAM IN COMPUTER SCIENCE

A Maturity Model based on ISO/IEC/IEEE 42010:2011 to Identify Technical Debt in Software Architecture

Master Thesis

Ademir Almeida da Costa Júnior

São Cristóvão – Sergipe

2020

FEDERAL UNIVERSITY OF SERGIPE

CENTER OF EXACT SCIENCES AND TECHNOLOGY

POSTGRADUATE PROGRAM IN COMPUTER SCIENCE

Ademir Almeida da Costa Júnior

A Maturity Model based on ISO/IEC/IEEE 42010:2011 to Identify Technical Debt in Software Architecture

Master Thesis presented to the Post-Graduation Program in Computer Science of the Federal University of Sergipe as a partial requirement to obtain a master's degree in Computer Science.

Advisor: Michel dos Santos Soares

São Cristóvão – Sergipe

2020

Ademir Almeida da Costa Júnior

A Maturity Model based on ISO/IEC/IEEE 42010:2011 to Identify Technical Debt in Software Architecture/ Ademir Almeida da Costa Júnior. – São Cristóvão – Sergipe, 2020-

83 p. : il. (algumas color.) ; 30 cm.

Supervisor: Michel dos Santos Soares

Master Thesis – FEDERAL UNIVERSITY OF SERGIPE
CENTER OF EXACT SCIENCES AND TECHNOLOGY
POSTGRADUATE PROGRAM IN COMPUTER SCIENCE, 2020.

1. Architecture Description. 2. Maturity Model. 3. ISO/IEC/IEEE 42010:2011. 4 Software Architecture. 5. Software Engineering. 6 Systematic Mapping Study. 7. Technical Debt. I. Michel dos Santos Soares. II. Federal University of Sergipe. III. Center of Exact Sciences and Technology. IV. Postgraduate Program in Computer Science. V. Título.

CDU 02:141:005.7

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-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: Ademir Almeida Da Costa Júnior

Em 19 dias do mês de novembro do ano de dois mil e dezenove, com início às 14h00min, realizou-se na Sala de Seminário do DCOMP da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Ademir Almeida da Costa Júnior**, que desenvolveu o trabalho intitulado: ***"A Maturity Model based on ISO / IEC / IEEE 42010: 2011 to Identify Technical Debt in Software Architecture"***, 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, Prof. Dr. **Rogério Patrício Chagas do Nascimento** (PROCC/UFS) e, em seguida, ao Prof. Dr. **Marcelo Medeiros Eler** (UFP). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) APROVADO "*(aprovado/reprovado)*". Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 25/2014/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"*, 19 de novembro de 2019.

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

[illegible]
Prof. Dr. Rogério Patrício Chagas do Nascimento
(PROCC/UFS)
Examinador Interno

[illegible]
Prof. Dr. Marcelo Medeiros Eler
(instituição)
Examinador Externo

Ademir Almeida da Costa Júnior
Ademir Almeida Da Costa Júnior
Candidato

I dedicate this work first to God, my wife, my precious daughter, and my parents. Also to my advisor, for his patience, advice and encouragement that have made it possible to complete this work.

Acknowledgements

To God, who gave me strength and inspiration to overcome all the challenges encountered during this walk. The advisor and friend Professor Michel dos Santos Soares, for guidance during the whole period of this dissertation, always with patience, perseverance and confident that I would reach my goal. To the professors of the Post-Graduate Program in Computer Science, for the advice, tips and patience that served as a stimulus here. To my parents, Ademir and Maria Lourdes. My daughter Zoe, for your smile and joy that stimulated me. To my wife Jocélia, for her unconditional support all the time, always giving me the courage to move on and achieve this goal. To the laboratory friends who, by means of talking, advice and exchanges of experiences contributed to this conquest. To the Federal University of Sergipe, and FAPITEC, sponsor, fundamental for the realization of the research project. I thank everyone who, directly or indirectly, helped me to my success in this work.

Once you stop learning, you start dying. (Albert Einsten)

Resumo

Arquitetura de software é considerada uma área importante da Engenharia de Software, pois é útil para gerenciar o desenvolvimento e a manutenção de sistemas intensivos de software em larga escala. A arquitetura de software como produto de desenvolvimento é útil para atividades técnicas, como descrever as visões e expectativas dos futuros produtos de software, bem como para atividades de gerenciamento, incluindo a alocação de tarefas para cada equipe e como entrada para as atividades de gerenciamento de projetos. Um problema principal ao descrever a arquitetura do software é saber quais elementos devem ser incluídos na arquitetura e em que nível de detalhe. Assim, a descrição de uma arquitetura de software é considerada uma entrega crucial em um processo de desenvolvimento de software, porque é lida por muitas partes interessadas em desenvolver e manter sistemas de software complexos compostos por vários elementos, incluindo software, sistemas, hardware e processos. Devido à importância da arquitetura de software, o padrão ISO/IEC/IEEE 42010:2011 foi publicado em 2011. Para facilitar e auxiliar na documentação da arquitetura de software, muitas contribuições foram propostas nas últimas décadas para os padrões de arquitetura, fornecidos pela academia e pela indústria. Esta dissertação de mestrado propõe o uso da norma ISO/IEC/IEEE 42010:2011 para desenvolver um modelo de maturidade, denominado ArchCaMo, baseado nas seções 5, 6 e 7 da norma mencionada. Para apoiar o projeto do ArchCaMo, foi realizado um Mapeamento Sistemático para descrever estudos que usavam explicitamente o padrão ISO/IEC/IEEE 42010:2011 e identificar quais partes desse padrão foram mais consideradas na literatura. ArchCaMo é útil para avaliar arquiteturas atuais e analisar a taxa de dívida técnica da arquitetura. Além disso, ele é eficaz para organizações que estão lutando para organizar, descrever e comunicar a arquitetura de software para vários interessados. Para cada nível de maturidade da arquitetura, a organização sabe o que esperar em relação a atividades e entregas. Dentro deste objetivo, três organizações são selecionadas como estudos de caso. Os pesquisadores conduziram a pesquisa por meio de entrevistas com seu arquiteto de software ou com o chefe da equipe de arquitetura de software. Ao analisar os resultados obtidos, os autores verificaram a conformidade de suas atividades de arquitetura de software com a norma ISO/IEC/IEEE 42010:2011. Como resultado, todas as três organizações foram classificadas no nível 1, o que significa que essas organizações falham em pelo menos um aspecto em formalizar e definir a arquitetura do software.

Palavras-chave: Descrição de Arquitetura. Modelo de Maturidade. ISO/IEC/IEEE 42010:2011. Arquitetura de Software. Engenharia de software. Mapeamento Sistemático. Dívida Técnica.

Abstract

Software architecture is considered an important area of Software Engineering, as it is useful for managing the development and maintenance of large scale software-intensive systems. The software architecture as a development product is useful for technical activities, such as describing the views and concerns of the future software products, as well as for management activities, including allocating tasks to each team and as an input for project management activities. One main issue when describing the software architecture is knowing what elements must be included in the architecture, and at what level of detail. Thus, the description of a Software Architecture has been considered a crucial deliverable in a software development process because it is read by many stakeholders when developing and maintaining complex software systems that are composed of multiple elements, including software, systems, hardware, and processes. Due to Software Architecture importance, the ISO/IEC/IEEE 42010:2011 standard was published in 2011. In order to facilitate and assist in the documentation of software architecture, many contributions have been proposed in the past decades for architectural standards, provided by academia and industry. This master thesis proposes the use of standard ISO/IEC/IEEE 42010:2011 to develop a maturity model, named ArchCaMo, which is based on sections 5, 6, and 7 of the mentioned standard. To support the designing of ArchCaMo, a Systematic Mapping Study was performed for describing studies that explicitly used the ISO/IEC/IEEE 42010:2011 standard, and identifying which parts of this standard were most considered in the literature. The ArchCaMo is useful to evaluate current architectures and analyze the rate of architecture debt. In addition, it is effective for organizations that are struggling with organizing, describing, and communicating the software architecture for multiple stakeholders. For each level of architecture maturity, the organization knows what to expect concerning activities and deliverables. Within this objective, three organizations are selected as case studies. The researchers conducted the survey by means of interviews with their software architect or the chief of the software architecture team. By analyzing the obtained results, the authors checked the compliance of their software architecture activities with ISO/IEC/IEEE 42010:2011. As a result, all three organizations were classified on level 1, which means that these organizations fail in at least one aspect to formalize and define the software architecture.

Keywords: Architecture Description. Maturity Model. ISO/IEC/IEEE 42010:2011. Software Architecture. Software Engineering. Systematic Mapping Study. Technical Debt

List of Figures

List of Tables

List of abbreviations and acronyms

ACM	Association for Computing Machinery
ACMM	Architecture Capability Maturity Model
ADL	Architecture Description Language
AF	Architecture Framework
ANSI	American National Standards Institute
ArchCaMo	Architecture Capability Model
BDUF	Big Design Up-Front;
CMMI	Capability Maturity Model Integration
DCOMP	Departamento de Computação
DoC	Department of Commerce
DTD	Documentation Technical Debt
EC	Exclusion Criteria
IC	Inclusion Criteria
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organization
IT	Information Technology
KAI	Key Architecture Item
RQ	Research Question
SMS	Systematic Mapping Study
SoaML	Service Oriented Architecture Modeling Language
SW-CMM	Capability Maturity Model for Software
SysML	Systems Modeling Language
TD	Technical Debt

TDM	Technical Debt Management
UML	Unified Modeling Language
UFS	Universidade Federal de Sergipe

Contents

1
Introduction

Software Architecture has been defined in multiple ways since the 1960s. From the first ideas of structuring and decomposing complex systems into more manageable modules (DIJKSTRA, 1968) (PARNAS, 1972b) (PARNAS, 1972a), to modern definitions with focus on cooperating components, decision making (TOFAN et al., 2014) and knowledge management (CAPILLA et al., 2016). Moreover, Software Architecture is considered an important area of Software Engineering, as it is useful for managing the development and maintenance of large scale software-intensive systems[1](FALESSI et al., 2010). The emphasis on mapping the components and their connectors of a software-intensive system is generally recognized and has led to better control over the design, development, and maintenance of these systems (BASS; CLEMENTS; KAZMAN, 2012).

Many researchers and practitioners have recognized software Architecture as a fundamental asset in software development and maintenance (GARLAN, 2000)(KRUCHTEN; OBBINK; STAFFORD, 2006) (GARLAN, 2014). For instance, the authors of a book on documenting architecture (BASS; CLEMENTS; KAZMAN, 2012) express that without an appropriate Software Architecture for the problem to be solved by software, the project will fail. Other authors also express the difficulties brought by not using a Software Architecture correctly to handle the complexity of software-intensive systems (OQUENDO, 2016) or the importance of Software Architectures for developing software systems that are better and more resilient to change when compared to systems developed without a clear architectural definition (BOOCH, 2007).

However, despite its importance, the process of software architecting, the evaluation of software architectures, and even the social impact on the software architecture in an organization are still neglected in software development (GARLAN, 2014) (BUCHGEHER; WEINREICH;

[1] In ("IEEE. . . , 2000) is defined as any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole" to encompass "individual applications, systems in the traditional sense, subsystems, systems of systems, product lines, product families, whole enterprises, and other aggregations of interest

KRIECHBAUM, 2016) (GALSTER; TAMBURRI; KAZMAN, 2017). Creating a strong, robust software architecture demands effort, which will be almost useless in case the architecture is not well-described and understood by technical stakeholders. Software architects need to describe it on the necessary level of detail, trying to solve ambiguity issues and organize it in such a way that it is understandable, and with information that is easy to retrieve.

In order to simplify the architect's work, the software architecture community has developed numerous technologies to support the architecture process (analysis, design, and review) (FALESSI et al., 2010). Besides, to facilitate and assist the software architecture documentation, several contributions have been made in the last decades in terms of architecture standards, for instance, Kruchten 4+1 View Model (KRUCHTEN, 1995), Siemens' 4 View Model (HOFMEISTER; NORD; SONI, 2000) and Zachman Framework (ZACHMAN, 1987).

By reading the ISO/IEC/IEEE 42010:2011 standard, many architectural elements in its chapters, including, to mention a few, architecture description, frameworks, viewpoints, relations, rationale, and also the relationship from this specific architectural standard to others are clearly described. For instance, the ISO/IEC 12207:2008 ("ISO/IEC. . . , 2008a), which establishes a common framework for software life cycle processes, and ISO/IEC 15288:2008 ("ISO/IEC. . . , 2008b), which establishes a common framework for describing the life cycle of systems created by humans. Therefore, from all these elements, there is no commonly known hierarchy, or at least a notion of minimal architecture, or even which parts most be considered by software architects. In summary, users do not know which elements are the most important ones, and as a result, some elements are barely mentioned in the literature. However, the issue here is that, in practice, it is hard to understand such standards, as they are considered too high-level to be used in practice or too complex to be easily understandable by the software development team (MOHAGHEGHI; APARICIO, 2017).

When the architectural documentation is insufficient, incomplete, or outdated, a case of technical debt occurs. One possible reason is due to the contest between agilists and more process-oriented personnel (BOOCH, 2007) (YANG; LIANG; AVGERIOU, 2016). Agilists often classify the efforts of defining an architecture as big design up-front (BDUF), which leads to massive documentation and implementation of features (HOFMEISTER; NORD; SONI, 2000). On the other hand, software architecture defenders consider that Software Architecture plays a vital role in achieving quality goals for large software-intensive systems (BOSCH, 2000). This especially applies to domains such as automobiles, telecommunications, finance, and medical devices (BOSCH, 2000).

The extension of software architecture documentation depends on the project's context, such as the environment, the domain, project size, stable architecture, business model, team distribution, rate of change, the system's age, and governance (Abrahamsson; Babar; Kruchten, 2010). In an agile project's, few architectural activities might be needed, but many large, complicated projects require significant architectural effort. For these projects, agile methods must suit the specific circumstances inherent in the development's context (Abrahamsson; Babar; Kruchten, 2010).

The assumption in this master thesis is that even though the importance of software architecture is recognized, how to describe, and at what level of detail one has to describe the software architecture, there are still open issues in the industry. Industry practitioners and even researchers frequently do not know how to start to describe the architecture (GARLAN, 2014), and how much architecture to use (WATERMAN; NOBLE; ALLAN, 2015)(WIRFS-BROCK; YODER; GUERRA, 2015). Also, they have issues regarding agility in processes and what is named Big Design Up-Front architecture (Abrahamsson; Babar; Kruchten, 2010), or even which architectural elements are necessary to describe in the software architecture documents (DING et al., 2014) (GRAAF et al., 2016) (DÍAZ-PACE et al., 2016).

1.1 Objectives

For this work, a general objective is defined, and it guided all the necessary efforts for its accomplishment. Notwithstanding, for this general objective to be effectively achieved, some specific objectives have to be progressively accomplished.

The general objective of this master thesis is to propose an evaluation of the adherence of software architectures using ISO/IEC/IEEE 42010:2011 as a criterion. Then, the Architecture Capability Model (ArchCaMo) to access the level of architectural maturity using ISO/IEC/IEEE 42010:2011 as the context is proposed. The objective of ArchCaMo is twofold. First, the ArchCaMo model can be used to evaluate current architectures, showing to development personnel at what level of maturity their architecture conforms to. Additionally, it is useful for those organizations that are struggling with organizing, describing, and communicating the software architecture for multiple stakeholders. For each level of architecture maturity, the organization knows what to expect in terms of activities and deliverables. Although other maturity models for software architectures were proposed, as described in the next section, no other models were introduced with a focus on ISO/IEC/IEEE 42010:2011.

Consequently, to achieve the general objective, the following specific objectives are proposed:

- Create a checklist by reading and analyzing the standard ISO/IEC/IEEE 42010:2011 to map all rules, guidelines, processes, instructions that are established in it;
- Propose a survey from the checklist, besides adding questions regarding the interviewee's identification;
- Apply the survey in public or private companies;
- Apply the survey in research papers that used the standard;

- Conduct a qualitative analysis based on data collected from the survey;
- Propose a maturity model to describe software architecture with the information obtained based on ISO/IEC/IEEE 42010:2011;
- Validate the maturity model employing data collected from the companies interviewed previously and the researched papers;

1.2 Methodology

The methodological instrumental definition must be directly related to the problem to be studied (LAKATOS; MARCONI, 2017). Thus, research must be rigorously analyzed even before its actual execution. Some procedures and techniques are defined to meet the general objective of this work, according to Table 1.

Table 1 – Summary of Methodological Approaches

Research's Classification by	
Objectives	Exploratory.
Variables' Nature	Qualitative
Technical Procedures	Bibliographic research; Systematic Mapping Study; Case Study
Perspective (Concentration Area)	Computer Science

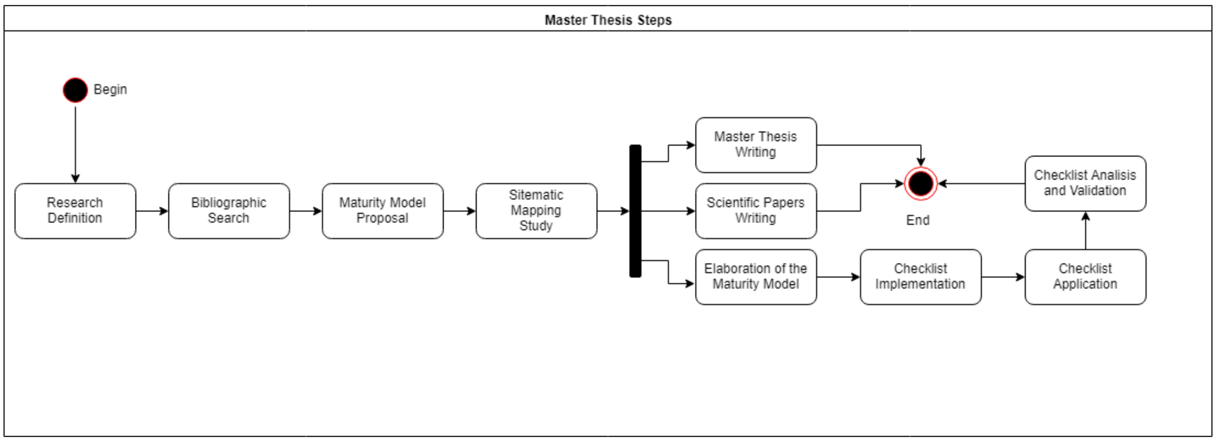
In terms of objectives, an exploratory research aims to provide more familiarity with the subject, aiming to make it more explicit or to establish hypotheses (GIL, 2002). This type of research aims at the improvement of ideas or the discovery of intuitions. Thus, this research can be classified as exploratory because it seeks to elaborate on a maturity model proposal for software architecture documentation.

As regards the technical procedures, Patton (2005) states that qualitative research implies three types of data collection: interviews, direct observations, and written documents. Interviews produce citations about people's experiences, opinions, feelings, and knowledge; data from observations consist of detailed descriptions of people's actions in the researched environment, and document analysis includes studying official publications, reports, and open-ended written responses to questionnaires and surveys (PATTON, 2005). According to Gil (2002), most of the time, research that has a more exploratory feature ends up taking the form of bibliographic research or case study. On the other hand, a Systematic Mapping Study offers and classifies an arrangement of the type of reports and research results that have been published, and usually provides a visual summary, the map, of its results (PETERSEN et al., 2008). A case study focuses on one instance of a phenomenon to be investigated, and it gives a detailed, in-depth description and insight of that instance. Additionally, complexity is fundamental to a case study to have success because it investigates multiple factors, events, and relationships that occur in a real-world case (JOHANNESSON; PERJONS, 2014).

Finally, according to SANTOS (2007), the research perspective represents the focus of interest. Therefore, this master thesis investigates the adherence of software architectures using ISO/IEC/IEEE 42010:2011 from the point of view of engineers, architects, and/or software analysts in the context of software development organizations in Sergipe, state of Brazil.

Figure 1 illustrates the process for this research, which contains ten developed activities.

Figure 1 – Master Thesis Activities.



1. Research Definition - Characterization of the research problem precisely in order to achieve cohesion between the study and the proposed problem. Thus, it provided a well-defined and detailed study reaching the proposed objectives.

2. Bibliographic Search - It consisted of identifying research in databases of scientific papers, thesis, dissertations, and related documents. This step helped the theoretical foundation of the present work.

3. Maturity Model Proposal - This activity aimed to propose a Maturity Model, named ArchCaMo, which intended to evaluate the documentation level of software architectures, using the ISO/IEC/IEEE 42010: 2011 standard as a basis.

4. Systematic Mapping Study - This activity aimed to identify works that used ISO/IEC/IEEE 42010:2011 for documenting their software architectures, and utilizing them to assist in the creation of the maturity model.

5. Proposal of the Maturity Model - It refers to how the maturity model is proposed employing the researchers' experience, ISO/IEC/IEEE 42010: 2011 standard, and bibliographic research.

6. Checklist implementation - At this stage, a checklist for validating the maturity model is prepared based entirely on ISO/IEC/IEEE 42010:2011.
7. Checklist Application - Step in which researchers personally applied the checklist in three different organizations.
8. Checklist Analysis and Evaluation - It aimed to achieve the proposed objectives according to a descriptive analysis concerning validation, based on the proposed maturity model.
9. Scientific Papers writing - Writing of scientific papers to submit in conferences, workshops, journals, or related venues.
10. Master Thesis Writing - It consisted of writing the master thesis. This research stage is carried out in parallel with activity 3.

1.3 Document's Organization

In addition to this introductory chapter, this master thesis presents six more chapters. These chapters are described as follows:

- Chapter 2 - Literature Review. It presents the conceptual basis for the development of this work, relating the three central areas of this research, Software Architecture, ISO/IEC/IEEE 42010:2011, Maturity models, and Technical debt.
- Chapter 3 - A Systematic Mapping Study on Software Architectures Description based on ISO/IEC/IEEE 42010:2011. It presents the process and results obtained from a systematic mapping study that searched state of the art about the use of ISO/IEC/IEEE 42010:2011 in academic works.
- Chapter 4 - ArchCaMo. It presents the proposed maturity model for software architecture description that is proposed in this master thesis.
- Chapter 5 - Evaluation of the Proposed Maturity Model. It presents the evaluation results of the proposed maturity model through three case studies.
- Chapter 6 - Conclusions and Future Works. It presents a summary of the objectives and methods applied, contributions, limitations, future works, difficulties and conclusions about this master thesis.

2 Literature Review

2.1 Software Architecture

Software Architecture aims to show and reflect adequately on the structure of the system, its constituent elements, and its relations (KOUROSHFAR et al., 2015). Furthermore, Software Architecture allows software engineers or architects to think about the functionalities and properties of a software system, without the need to revise source code and/or implementation details (PERRY; WOLF, 1992).

Bosch (2004) states that Software Architecture should be modeled and represented as the composition of a set of architectural design decisions regarding domain models, architectural solutions, variation points, features, and scenarios needed to meet the requirements elicited using Stakeholders. Additionally, Software Architecture means the structure or structures of a system, which includes essential features, their externally-visible behavior, and the relationships among them (BASS; CLEMENTS; KAZMAN, 2012)

There are several definitions for Software Architecture, so there is no homogeneous consensus in the literature (KRUCHTEN; OBBINK; STAFFORD, 2006). However, it is known that Software Architecture is not the operational software, but rather, a representation that allows analyzing the effectiveness of the project to satisfy the declared requirements (PRESSMAN; MAXIM, 2014). Then, the architecture of a system is a set of decisions, so it eases or constrains the achievement of the desired system properties. Examples of decisions are the type of programming language to adopt, and the database for data persistence (FALESSI et al., 2011).

Conforming to Ding e Medvidovic (2001), every software system has a Software Architecture, which can be implicit, not documented, or explicit, that is, documented and specifically designed to achieve predetermined business goals and quality requirements. As a result, Software Architecture is one of the first phases of the software development process, so it significantly constrains and facilitates the achievement of requirements and business goals

(MARANZANO et al., 2005).

Software Architecture not only establishes how the system should be designed, but it also manages the system's evolution, then stability and flexibility should be considered while designing the architecture (JAZAYERI; RAN; LINDEN, 2000). Furthermore, when a software system needs to be updated, it may be necessary to spend time documenting the underlying software architecture even if it has to be recovered (reverse engineered) (DING; MEDVIDOVIC, 2001). Hence, reviewing the software architecture documentation represents a valid effort to check if the system is in conformance with the business goals, and to reveal any potentially-missed objectives early on (MARANZANO et al., 2005).

2.2 ISO/IEC/IEEE 42010:2011

The standard ISO/IEC/IEEE 42010:2011 (ISO/IEC/IEEE, 2011) addresses the creation, analysis, and sustainment of architectures of software systems through the use of architecture descriptions. This standard proposes a conceptual model of architecture description, including concepts such as views, concerns, models, rationale, frameworks, architecture description languages, and viewpoints. Furthermore, the standard establishes a core ontology for describing and evaluating software architectures, which is useful for stakeholders when they need to establish the elements of software architecture. Also, it provides a terminological framework for discussions on architectural descriptions (ISO/IEC/IEEE, 2011) within the software development team.

ISO/IEC/IEEE 42010:2011 is the updated standard that cancels and replaces ISO/IEC 42010:2007 [1], which is a version with improvements of former ANSI/IEEE 1471-2000. The original IEEE 1471 specified requirements on the contents of architecture descriptions of systems. ISO adopted the IEEE standard in 2007 as ISO/IEC 42010. Later, ISO and IEEE produced a joint revision, resulting in the ISO/IEC/IEEE 42010, System and Software Engineering — Architecture Description. ISO/IEC/IEEE 42010:2011 will be replaced by ISO/IEC/IEEE AWI 42010 Software, Systems and Enterprise – Architecture description, which is currently under development.

Architectural concepts, principles, and procedures are being applied and revised to help manage the increasing complexity faced by system stakeholders. Several terms that have some relation to software architecture are defined in the standard. Those considered important are defined bellow:

1. *Architecting* is the process of conceiving, expressing, defining, documenting, communicating and certifying proper implementation of, maintaining and improving an architecture throughout a software's life cycle.
2. An *Architecture* is the fundamental concept or properties of a system in its environment embodied in its elements, relationships, and the principles of its design and evolution.

3. A *System* is defined as a combination of interacting elements organized to achieve one or more stated purposes, and it is situated in an environment.

4. An *Environment* is a context determining the totality of influences upon the system, including its interactions with that environment. The environment is understood through the identification and the analysis of the system's stakeholders and their concerns.

5. The *Stakeholders* of a system are parties with interests in that system, and their interests are expressed as concerns.

6. A *Concern* could be held by one or more stakeholders. Concerns arise throughout the life cycle from system needs and requirements, from design choices and from implementation and operating considerations.

7. *Architecture Descriptions* are work products of systems and software architecting. An architecture description includes one or more architecture views.

8. An *Architecture View* (or simply, view) addresses one or more of the concerns held by the system's stakeholders. An architecture view expresses the architecture of the system-of-interest in accordance with an architecture viewpoint (or simply, viewpoint).

9. An *Architecture Viewpoint* is an work product that establish the conventions for the construction, interpretation and use of architecture views to frame specific system concerns.

10. *Architecture View* is an work product expressing the architecture of a system from the perspective of specific system concerns. An architecture view is composed of one or more architecture models.

11. A *Model Kind* is a convention for a type of modelling, as data flow diagrams, class diagrams, Petri nets, balance sheets, organization charts and state transition models.

12. An *Architecture Rationale* records explanation, justification or reasoning about architecture decisions that have been made. The rationale for a decision can include the basis for a decision, alternatives and trade-offs considered, potential consequences of the decision and citations to sources of additional information.

13. An *Architecture Framework* establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community.

14. An *Architecture Description Language* (ADL) is any form of expression for use in architecture descriptions. An ADL provides one or more model kinds as a means to frame some concerns for its audience of stakeholders.

ISO/IEC/IEEE (2011) does not specify any format for recording architecture descriptions, but it describes the basic context of an architecture description. Besides, the standard specifies the best practices for documenting enterprise, system and software architectures.

By considering the terms in Figure 2, one can notice that the context specifies that stakeholders have interests in one or more software systems. These interests, better explained as concerns, include a variety of extra-functional properties, such as maintainability, testability, and modularity, but also project management concerns, including costs, schedule, business goals, and strategies. A concern pertains to any influence on a system in its environment. Each system is situated in an environment, which is a context determining the setting and circumstances of all influences upon a software system. The environment of a software system includes developmental, business, technological, operational, organizational, political, economic, regulatory, legal, ecological, and social influences.

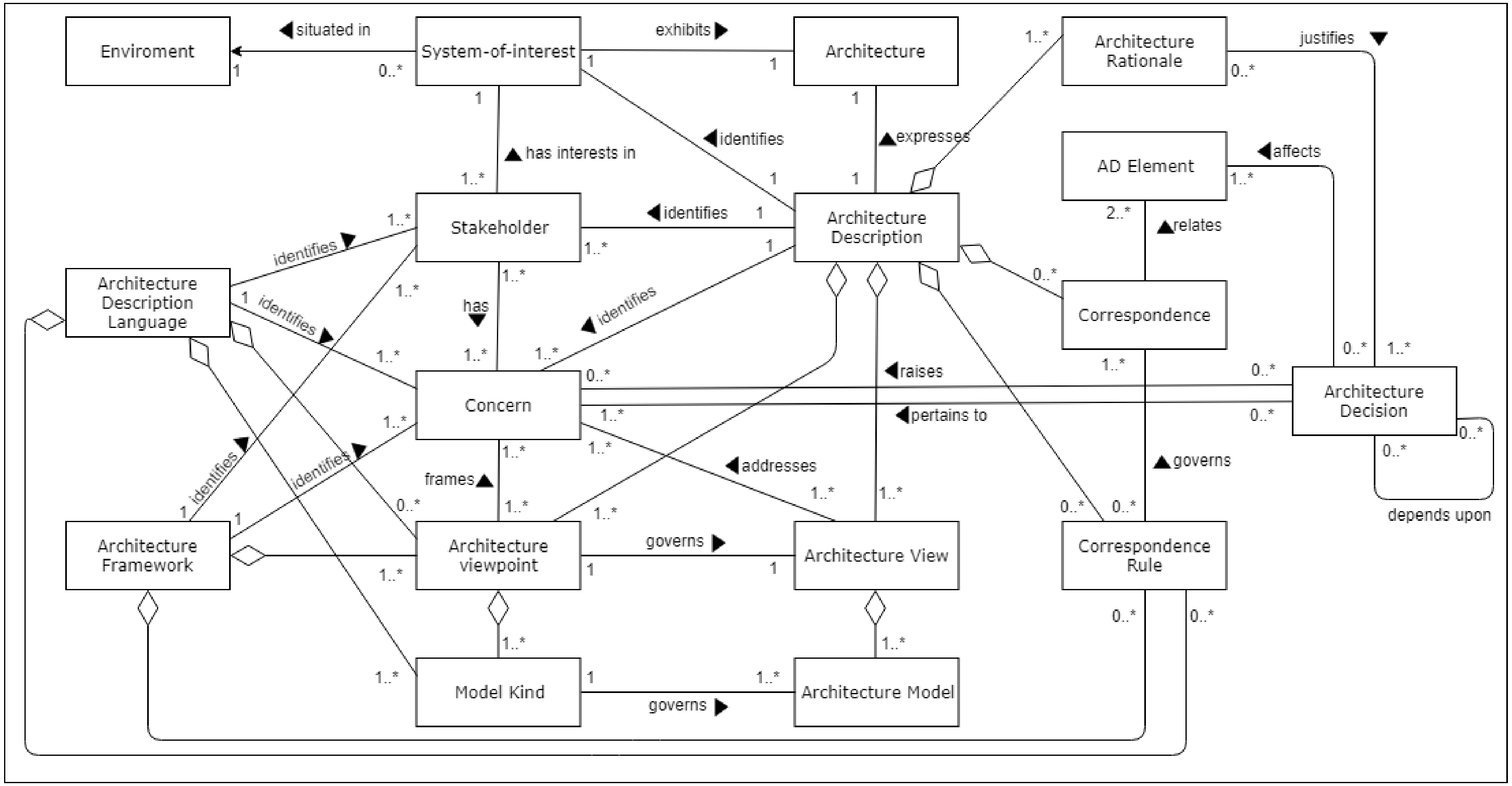
Each class of stakeholder has more or less interest, depending on how important a concern is regarding its tasks and roles in the organization. For instance, maintainability is a concern of software developers and architects, and business goals and strategies are concerns for managers. Each system can exhibit many architectures, for instance, when considered in different environments.

An architecture can be expressed through several distinct architecture descriptions, and the same architecture can characterize one or more software systems, as a software product line sharing a common architecture.

Architecture descriptions have many uses for a variety of stakeholders throughout the system life cycle. For instance, software developers use the architecture description for software design, development, and maintenance activities. Clients, acquirers, suppliers, and developers use the architecture description as part of contract negotiations, documenting the characteristics, features, and design of a software system. Infrastructure personnel uses the architecture description as a guide to operational and infrastructure support and configuration management. Managers use the architecture description as support to software planning activities, such as establishing the schedule, budget, and the team.

A complete description of ISO/IEC/IEEE 42010:2011 can be find in the official standard document (ISO/IEC/IEEE, 2011), and its practical use has been explored by many researchers in past years, for instance in health systems (CRICHTON et al., 2012)(FRANÇA; LIMA; SOARES, 2017), industry (HEESCH; AVGERIOU; HILLIARD, 2012) (MUSIL et al., 2015), transportation systems (KARKHANIS; BRAND; RAJKARNIKAR, 2018a), business (VIDONI; VECCHIETTI, 2016), defense (WILLIAMS; STRACENER, 2013) and energy (EFFENBERGER; HILBERT, 2016).

Figure 2 – Conceptual model of an architecture description.



Source: Adapted from (ISO/IEC/IEEE, 2011)

2.3 Maturity Models

Making a software product with a high level of quality that also meets the financial plan and agenda is the main goal of any organization. This task usually implies making trade-offs among conflicting aspects, for example, number of features to implement, time-to-market, quality for the user, and system maintenance throughout its life cycle (FALESSI et al., 2010).

In general, the term "maturity" refers to a "state of being complete, perfect, or ready" (ONLINE, 2019) and implies some progress in the development of a system. Accordingly, maturing systems (e.g. biological, organizational or technological) increase their capabilities over time regarding the achievement of some desirable future state. Thus maturity can be conquered qualitatively or quantitatively in a discrete or constant approach (SCHUMACHER; EROL; SIHN, 2016).

Maturity models are commonly used as an instrument to conceptualize and measure the maturity of an organization or a process regarding some specific target state (SCHUMACHER; EROL; SIHN, 2016). Many maturity models were proposed in the past decades to access the capability of organizations and departments of information technology. Among these, SW-CMM and CMMI are well-known and have been applied to evaluate software development processes in many countries for a variety of domains.

SW-CMM (PAULK et al., 1993), and later its evolution, Staged CMMI (CHRISSIS; KONRAD; SHRUM, 2011a), propose evaluation in 5 levels, in which each level includes several activities, tasks, goals, processes, practices and so on. Therefore, when an organization is certified at some level, it is possible to know which activities and practices are performed, indicating a level of discipline and organization for software development.

Few works have proposed to evaluate and access maturity levels of software architectures. For instance, in (AHMED; CAPRETZ, 2011), authors contribute towards the establishment of a comprehensive and unified strategy for process maturity evaluation of software product lines, showing the maturity of the architecture development process in two organizations.

Authors of article Ostadzadeh e Shams (2014) propose an architectural maturity model framework to improve Ultra-Large-Scale Systems Interoperability. Although the authors present an architectural maturity model, their focus was only on organizing an interoperability maturity model, i.e., the maturity is evaluated from the interoperability point of view, and is not applicable for other architectural purposes.

The US Department of Commerce (DoC) has developed an IT Architecture Capability Maturity Model (ACMM) (MEYER; HELFERT; O'BRIEN, 2011). The ACMM provides a framework that represents the key components of a productive IT architecture process. Their approach is to identify weak areas and provide an evolutionary path to improving the overall architecture process. The DoC ACMM consists of six levels and nine architecture characteristics.

To the best of our knowledge, no works have described a software architecture maturity model based on ISO/IEC/IEEE 42010:2011, a standard to describe software and system architectures. Therefore, the novelty here regards not only the proposal of a new architectural capability model for evaluation of software and systems architectures, but also the use of a standard as a guide to the capability model.

The process of continuous improvement in an organization is a differential strategy when applied as a maturity model that contains and unifies the practices used in specific disciplines such as Software Engineering, Software Architecture, Product Development, and Integrated Processes (CHRISSIS; KONRAD; SHRUM, 2011b).

Maturity models are useful for understanding the organization's current level against a standard set of definitions, and these models help the organization understand inherited consequences, and what business objectives are achieved by moving to a higher level of maturity (KREGER et al., 2009). Moreover, through maturity models, it is possible to evaluate organizational performance, supporting management, and allowing improvement (MAIER; MOULTRIE; CLARKSON, 2012).

Maturity modeling is a generic approach that describes an organization's development of time progression through optimal levels to an end state (KLIMKO, 2001). Then, maturity models are instruments used to evaluate organizational elements and select appropriate actions that lead to higher levels of maturity (KOHLEGGER; MAIER; THALMANN, 2009).

In terms of Software Architecture Description, its improvement by the parties, in the way that software architectures are represented/documented, becomes important as it can enhance communication and cooperation, management, as well as evaluation and maintenance of these architectures, enabling to work in an integrated and coherent fashion (ISO/IEC/IEEE, 2011)(GUESSI et al., 2015).

2.4 Technical Debt

Technical debt (TD) is a metaphor used to describe a technical problem on software development, which regards how to balance a short-term benefit with long-term quality of a software system (LI; AVGERIOU; LIANG, 2015)(ERNST et al., 2015). The idea of TD is strongly associated with time, and precisely the metaphor points out the short-term gain given by a sub-optimal solution against the long-term one considered optimal. Sometimes, this metaphor may be practical for estimating if a technical solution is sub-optimal or might be optimal from the business point of view (MARTINI; BOSCH; CHAUDRON, 2015).

Similar to financial debt, TD provokes interest payments, which requires additional work that the development team has to dedicate in the future because of previous design choices (KRUCHTEN et al., 2013). Therefore, the team has to choose to continue paying the interest, or pay down the principal by refactoring the actual design into a better one (KRUCHTEN et al., 2013).

TD is unavoidable, and can even be a good thing, as long as it is managed properly, for instance, documenting a system briefly, but adding all the missing information before delivery the system (ALLMAN, 2012). In (LIM; TAKSANDE; SEAMAN, 2012), it was reported that project teams recognize that technical debt is inevitable and necessary within business realities. Moreover, TD comes from multiple causes, its consequences are hard to predict, and usually involves speculation about what will occur in the future (ALLMAN, 2012). Although TD benefits, there is a high cost in the future because immature artifacts, such as poor design, incomplete documentation, and unfinished testing, may turn the system more complex, less understandable, and thus need extra effort to modify (SHULL, 2011).

Probably, the term TD was first mentioned at the source code level, as reported in (CUNNINGHAM, 1992), which describes a portfolio management system that provides basic accounting, record keeping, and reporting, as well as analytical computations to assist the manager of cash portfolios. Later on, TD extended from source code level to other software products as software architecture, detailed design, documentation, requirements, and testing (BROWN et al., 2010).

One classification used for TD, popularly adopted in academia (KRUCHTEN; NORD; OZKAYA, 2012)(ALVES et al., 2014)(LI; AVGERIOU; LIANG, 2015), is Steve McConnell's definition, which divides it into intentional and unintentional accumulation of technical debt (MCCONNELL, 2008). TD that occurs unintentionally happens, for instance, when an inexperienced programmer writes bad code. In other words, this technical debt is the result of doing a poor job, or it can be incurred unknowingly. The second one, the intentional, is a result of conscious decisions to optimize for the present rather than for the future (HOLVITIE et al., 2018).

TD management (TDM) requires actions that will prevent potential TD (both intentional and unintentional) from being incurred, and these actions need to deal with the TD which is already accumulated to make it visible and controllable, and to maintain a balance between cost and value of the software project (LI; AVGERIOU; LIANG, 2015). Additionally, TDM requires the commitment of a team to the project, and the fact to accept some technical risks to achieve business goals (LIM; TAKSANDE; SEAMAN, 2012).

In Li, Avgeriou e Liang (2015), the authors performed a systematic mapping study to identify and analyze research on TD and its management, covering publications between 1992 and 2013, so they categorize ten types of TD in software development: Requirements TD, Architectural TD, Design TD, Source Code TD, Test TD, Build TD, Documentation TD, Infrastructure TD, Versioning TD, and Defect TD.

2.4.1 Architectural Technical Debt

The Architectural smells is a consequence of using architectural decisions that negatively impact system quality, but it does not always happen intentionally. Its motivation may be caused by applying a design solution in an inappropriate context, mixing design fragments that have undesirable emergent behaviors, or applying design abstractions at the wrong level of granularity (GARCIA et al., 2009a).

Architectural TD (ALVES et al., 2014) involves issues detected in project architecture, for instance, architecture diagrams not in conformance with source code, and a developer that makes a poor design decision to solve a short term problem at the expense of a more robust solution. Usually, these types of debts cannot be solved only in source code, but in the system's properties and its concepts (KRUCHTEN; NORD; OZKAYA, 2012).

According to Lippert e Roock (2006), (GARCIA et al., 2009b) and Fontana, Ferme e Zanoni (2015), Architectural Smells are generally adopted architectural decisions that negatively impact system quality and are caused by a violation of recognized design principles. Architecture smells indicate areas in the system's architecture that should be further examined (FONTANA; FERME; ZANONI, 2015). Furthermore, it precisely changes lifecycle properties, but it also can have side effects on other quality properties such as performance and reliability (GARCIA et al., 2009a). Besides, some source code smells can also indicate architecture degradation, called architecturally relevant code anomalies (MACIA et al., 2012).

When a technical debt needs to be repaid, maintenance and evolution challenges are the tasks to be made, particularly when the payback involves refactoring and re-architecting (NORD et al., 2012). Similarly, Architectural smells are remedied by altering the internal structure of the system and the behaviors of internal system elements without changing the external behavior (GARCIA et al., 2009a). Furthermore, in consonance with (LEHMAN; BELADY, 1985), changing will expand software complexity, leading to software decay if refactoring is not performed as needed. As a result, identifying and repairing code smells may improve system maintainability, but many maintainability problems derive from poor use of software architecture-level abstractions, as components, connectors and styles, rather than implementation level (GARCIA et al., 2009a).

2.4.2 Documentation Technical Debt

As reported in Li, Avgeriou e Liang (2015), Documentation TD (DTD) refers to insufficient, incomplete, or outdated documentation in any aspect of software development, as out-of-date architecture documentation or a lack of code comments. Documentation is not only a document containing the software specifications, but the code comments as well (SEAMAN; GUO, 2011). This type of debt is added each time when a piece of software is altered/updated, but the documentation is not modified according to the alterations made (SEAMAN; GUO, 2011).

Moreover, as stated in (GUO; SEAMAN, 2011), DTD is identified by comparing change reports to documentation version histories.

Insufficient documentation may not express a “defect” in the purpose that will lead a software with errors, but lack of documenting can represent DTD for other teams that prioritize reuse and maintainability (SHULL, 2011). In Falessi et al. (2010), the authors point out that little documentation leads to high maintenance effort, and it is challenging to estimate the cost (interest) of not documenting design in UML or, commenting a Java class, or capturing design rationale.

In Seaman e Guo (2011), the authors classify TD’s that are automatically detectable. Then, DTD is a type of TD which is not automatically detectable, and a human inspector can only discover these types of smells. For example, the quality (not the amount) of documentation; a computer algorithm can not determine if the documentation is simple to understand, or if it is according to the implemented code statements (SEAMAN; GUO, 2011).

3 A Systematic Mapping Study on Software Architectures Description based on ISO/IEC/IEEE 42010:2011

This chapter presents the design and execution of a Systematic Mapping Study (SMS), which follows procedures described in (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). A SMS provides a structure of the type of research reports and results that have been published by classifying them and often provides a visual summary, the map of its results (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Besides, this chapter presents the process for conducting the SMS, which is composed of Research Question (RQ), Research String, Data Source Selection, and Selection of Primary Studies. Therefore, all these steps were considered to understand the use of ISO/IEC/IEEE 42010:2011 regarding its most used items.

3.1 Research Questions

The following research question is proposed: "What studies in the literature have used ISO/IEC/IEEE 42010:2011 as a reference for designing their software architectures?". In order to answer this research question, a set of questions is defined, as displayed in Table 2.

Table 2 – Research Questions

Abbreviation	**Question**
Q1	When was the ISO/IEC/IEEE 42010:2011 used?
Q2	What was the venue of publication?
Q3	What aspects from ISO/IEC/IEEE 42010:2011 are most considered?
Q3.1	Which models were used?
Q4	In which domains were ISO/IEC/IEEE 42010:2011 used?
Q5	What type of validation was considered?
Q6	What type of research strategy was considered?

The domains cited in Question Q4 are the main final activities to which the software architecture will be applied in software and systems development, as for instance, Education, Transportation, Communication, and so on.

Validating research is to demonstrate to the reader that the result of the study is valid. Identification of the possible types of validation mentioned in Question Q5 are those proposed in (SHAW, 2003): Analysis, Experience, Evaluation, Example, Persuasion, and Blatant assertion.

A research strategy is a method for conducting a research study, conducting a research in planning, executing, and monitoring the study (JOHANNESSON; PERJONS, 2014). Types of research strategies mentioned in question Q6 are those presented in (JOHANNESSON; PERJONS, 2014): Experiments, Surveys, Case Studies, Ethnography, Grounded Theory, Action Research, Phenomenology, Simulation, Mathematical and Logical Proof.

3.2 Search Strategy

A test research was performed to prepare the generic search string and to choose the databases of papers. First of all, the authors searched on IEEE Xplore Digital Library, using the string “ISO 42010”. Then, further words of other papers were included, resulting in the following keywords: ISO 42010, ISO/IEC 42010, IEEE 42010, ISO-IEC-IEEE 42010, ISO/IEC/IEEE 42010, ISO STANDARD 42010. Then, at the end of this stage, the following bibliographic bases were chosen: ACM Digital Library, IEEE Xplore, Science Direct, Scopus, and Web Of Science.

The defined generic search string is displayed in Table 3.

Table 3 – Generic search string.

(**TITLE-ABS-KEY** (“ISO 42010” OR “ISO/IEC 42010” OR “ISO/IEC/IEEE 42010” OR “ISO-IEC-IEEE 42010” OR “IEEE 42010” OR “ISO STANDARD 42010”))

Since the first draft of ISO/IEC/IEE 42010:2011 was published in 2007, papers with a date of publication prior to this year were discarded. Also, the inclusion and exclusion criteria were defined to identify relevant works to this research, which are presented as follows:

Inclusion criteria (IC):

- IC1 - Studies that use ISO/IEC/IEEE 42010:2011 for defining, evaluating, describing, or developing software architectures.

Exclusion criteria (EC):

- EC1 - Duplicated papers;
- EC2 - Secondary study papers;

- EC3 - Papers written in languages other than English;
- EC4 - Posters;
- EC5 - Standards;

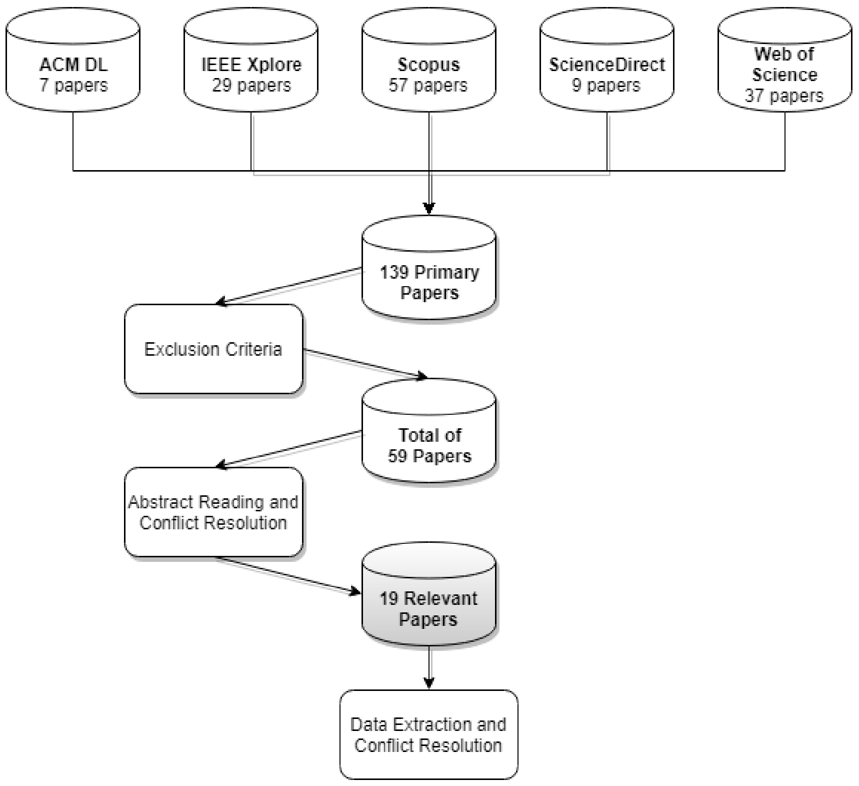
The search was conducted two times, the first one on September 12, 2018, returning 128 papers, and the second search, complementing the first one, was conducted on June 07, 2019, returning 11 additional papers, performing a total of 139 papers. This was made to keep this research most updated as possible.

The results from the databases were exported via *bibtex* file, and imported into tool StArt(LAPES, 2018), which assists in performing Systematic Literature Reviews. This tool is used to organize the references of articles, and then all the references were exported to an online spreadsheet, in this case, Google Sheets.

3.3 Results and analysis

Figure 3 describes the organization of the SMS's steps, and the number of studies resulted in each one. First, using the adopted support tool and manual analysis, 80 papers were rejected regarding the defined Exclusion criteria.

Figure 3 – Representation flow of the SMS.



Second, a selection step was performed. In order to avoid bias, all papers were analyzed independently by two researchers, who applied the defined inclusion, and exclusion criteria by reading the title, the keywords and the abstract. At the end of this stage, a meeting was held to resolve doubts or conflicts from the selection, resulting in the final list of 19 papers, 7 from IEEE Xplore, 4 from ScienceDirect, 4 from WebOfScience, 3 from Scopus and 1 from ACM Digital Library.

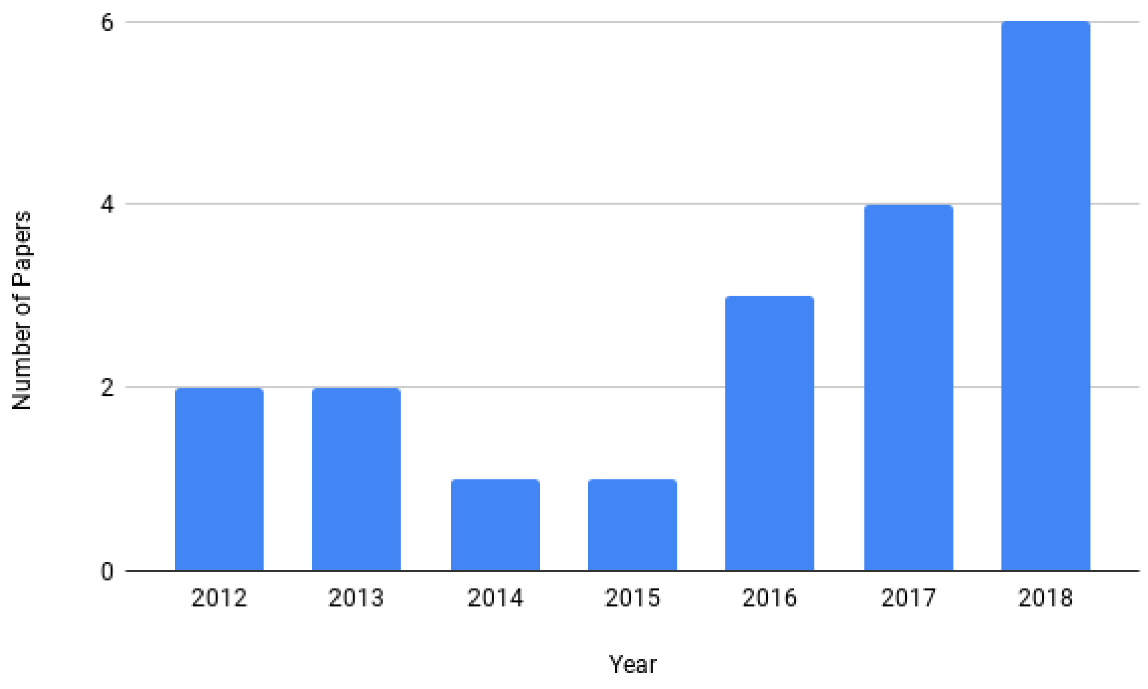
Finally, data were extracted from the papers in order to answer the questions defined in Table 2. In addition, in the selection phase, data extraction was performed individually by each researcher. Then, at the end of that phase, a meeting was held again to solve doubts or conflicts.

3.3.1 Q1 - When was ISO/IEC/IEEE 42010:2011 used?

When the 19 papers were analyzed, it was found that the years of 2016, 2017, and 2018 have the majority of papers, three, four, and six, respectively, as depicted in Figure 4. On the other hand, the years that have fewer papers are 2014 and 2015.

Even though the first draft of the standard was released in 2007, there is a gap from 2007 until 2011, the final version, when no paper was selected.

Figure 4 – Q1 - Paper's Distribution.



3.3.2 Q2 - What was the venue of publication?

The purpose of this research question was to evaluate from what sources these papers came from. The majority of papers was published in conferences, 13 papers, and the remainder was published in journals, 6 papers. This result may indicate that researchers in Computer Science, specifically in Software Engineering, prefer to publish in conferences, or that the papers are still not mature and with enough details to be published in journals. Additionally, there are 17 different venues of publication, with only one mentioned twice: Journal of Systems and Software.

This is an important finding of our research, because it allows other researchers to know in which venues articles about ISO/IEC/IEEE 42010:2011 have been published.

3.3.3 Q3 - What aspects from ISO/IEC/IEEE 42010:2011 are most considered?

The purpose of this research question is to identify the most used points of the standard ISO/IEC/IEEE 42010:2011 by the researchers. As a research paper most often has a limited number of pages, it was inferred that the researchers may not write all the possible aspects from the standard. It was listed the main ones to be checked on each paper, which includes: Stakeholders (St), Concerns (Cn), View (Vi), Viewpoint (Vp), Model (Md), Model Kind (MK), Correspondence (Co), Correspondence Rule (CR) and Rationale (Rt), showed in Table 4.

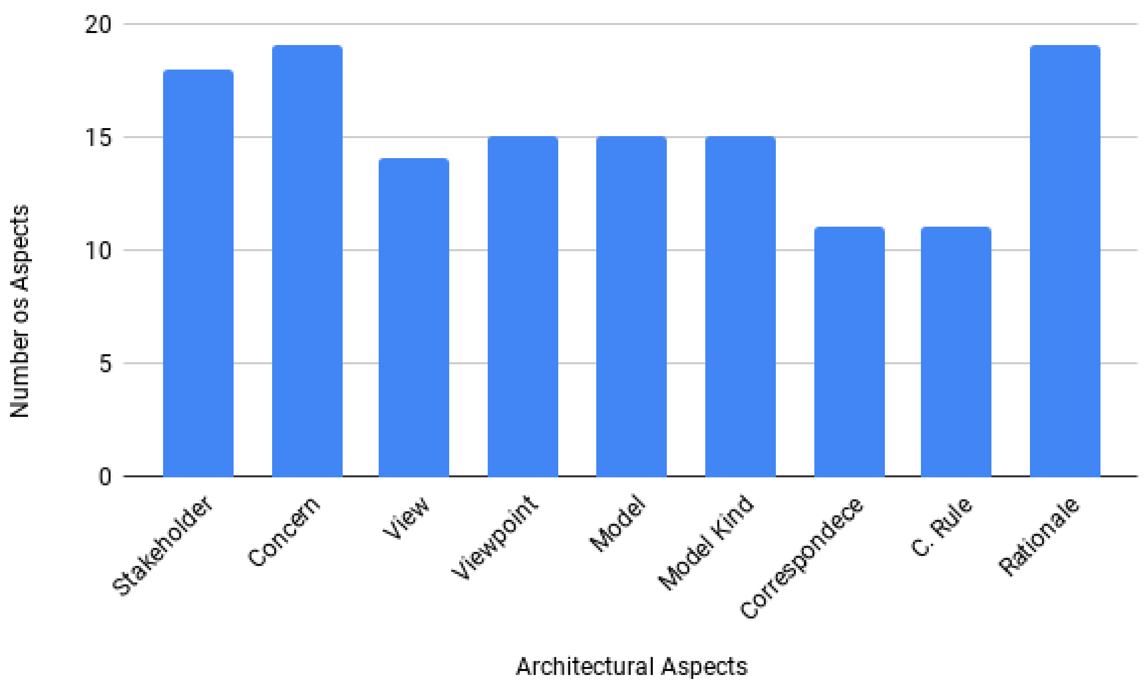
Table 4 – Aspects of ISO/IEC/IEEE 42010:2011 found in the papers.

Paper	**St**	**Cn**	**Vi**	**Vp**	**Md**	**MK**	**Co**	**CR**	**Rt**
(KANNENGIESSER; MüLLER, 2018)	•	•	•	•	•	•	•	•	•
(KARKHANIS; BRAND; RAJKARNIKAR, 2018b)	•	•	•	•	•	•			•
(AMIN; BLACKBURN; GARSTENAUER, 2018)	•	•					•	•	•
(KAVAKLI et al., 2018)	•	•	•	•	•	•	•	•	•
(OBERGFELL et al., 2018)	•	•	•	•	•	•			•
(MO; BECKETT, 2018)	•	•		•					•
(FRANÇA; LIMA; SOARES, 2017)	•	•	•		•	•	•		•
(CHAABANE; BOUASSIDA; JMAIEL, 2017)	•	•		•	•	•			•
(GÓMEZ et al., 2017)	•	•	•		•	•	•	•	•
(MAY et al., 2017)	•	•	•	•	•	•			•
(DAS, 2016)	•	•	•	•	•	•	•	•	•
(EFFENBERGER; HILBERT, 2016)	•	•	•	•					•
(VIDONI; VECCHIETTI, 2016)		•	•	•	•	•	•	•	•
(MUSIL et al., 2015)	•	•	•	•	•	•	•	•	•
(PANUNZIO; VARDANEGA, 2014)	•	•	•	•	•	•			•
(WILLIAMS; STRACENER, 2013)	•	•		•	•	•	•	•	•
(HILLIARD et al., 2012)	•	•	•	•	•	•	•	•	•
(CRICHTON et al., 2012)	•	•	•				•	•	•
(HEESCH; AVGERIOU; HILLIARD, 2012)	•	•		•	•	•	•	•	•

These aspects may take a considerable area in the selected papers because they require explanation. Thus, it is possible that the authors preferred to omit them, and not necessarily that they were not considered in their research.

The most used aspects are Concerns and Rationales, mentioned by all papers, as depicted in Figure 5. When defining the stakeholders, concerns came as a consequence. There is one paper, (VIDONI; VECCHIETTI, 2016), that did not explicitly mentioned stakeholders, and all the requirements were extracted from academic literature. Rationales are presented in any software architecture because they are the reason for decisions.

Figure 5 – Q3 - Most considered Architectural Aspects of ISO/IEC/IEEE 42010:2011.



Stakeholders are the key to any architecture, as they define the main elements of a Software Architecture, 18 papers mentioned this aspect.

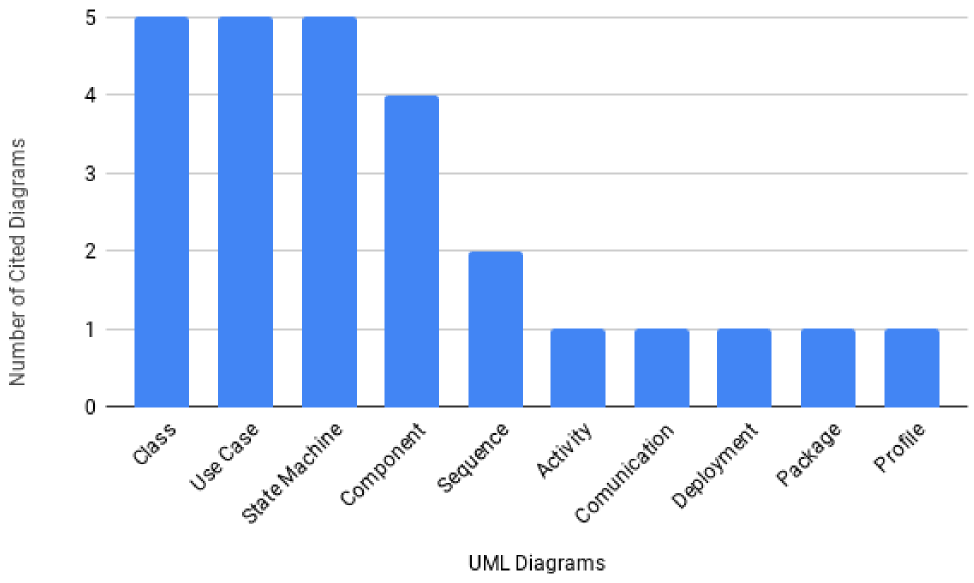
The third most mentioned architectural element is Model and Model Kind, 15 out of 19 papers mentioned them. Model and Model Kind refers basically to the type of diagrams, and the most used modeling language is Unified Modeling Language (UML), in 10 papers, followed by SysML (3 papers) and SoaML (1 paper). One paper mentioned both UML and SysML, and another used UML and SoaML. This result may be expected because UML is probably the most used modeling language in Software Engineering (HUTCHINSON; WHITTLE; ROUNCEFIELD, 2014)(CHAUDRON, 2017).

Another finding of this SMS is that some of the papers that identified the views did not mention what views they used. In contrast, others explicitly mention what views they used: (CRICHTON et al., 2012), (PANUNZIO; VARDANEGA, 2014), (VIDONI; VECCHIETTI, 2016), (EFFENBERGER; HILBERT, 2016), (MAY et al., 2017), (FRANÇA; LIMA; SOARES, 2017), (GÓMEZ et al., 2017), (KARKHANIS; BRAND; RAJKARNIKAR, 2018b) and (KAVAKLI et al., 2018).

3.3.3.1 Q3.1- Which models were used?

When analyzing the Model Kinds used in the papers, the Use Case, Class, and State Machine diagrams are the most mentioned ones, as shown in Figure 6. This may be an expected result because Use Case diagrams describe scenarios and functional requirements (BERTOLINO et al., 2002), making it easier to understand what the system does and give a reasonable means of communication about the system (JOHN; MUTHIG, 2002). Furthermore, Class Diagrams can be directly mapped to object-oriented languages, and they are the basis of software construction. The Component diagram is the fourth most mentioned model, in four papers, followed by the Sequence diagram, 2 times. The remaining models were used only in one paper.

Figure 6 – Q3.1 - UML diagrams that were mentioned on the architectures.



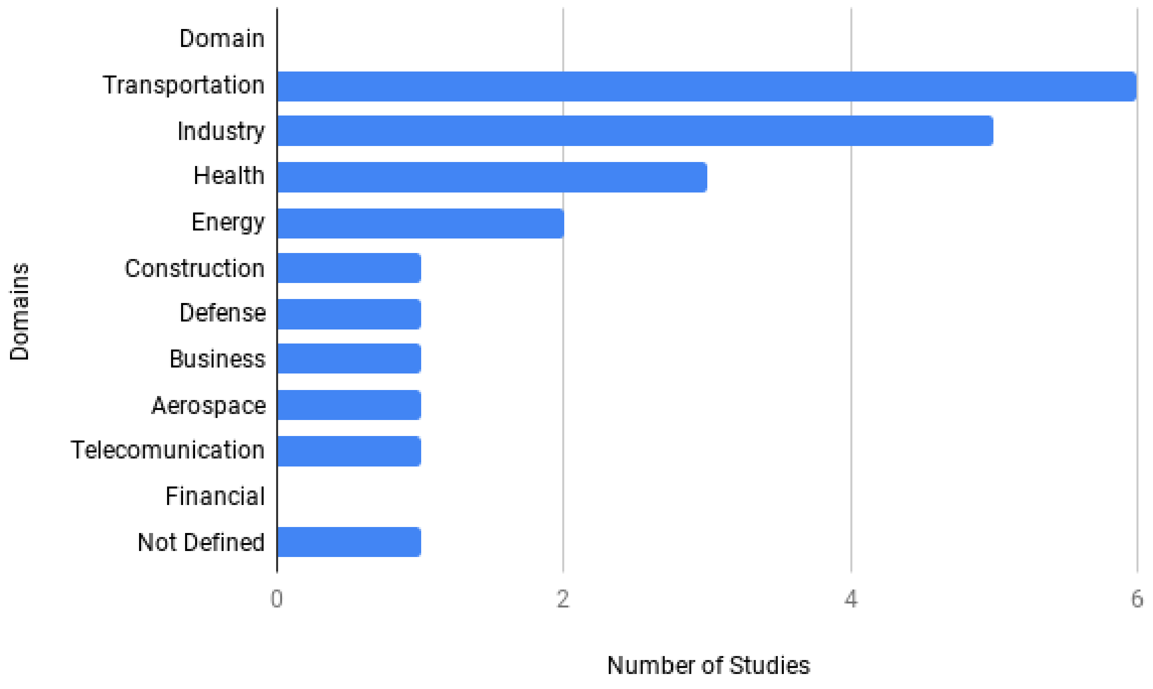
In paper (KARKHANIS; BRAND; RAJKARNIKAR, 2018b), Definition Block, and Internal Block diagrams were mentioned. Publication (WILLIAMS; STRACENER, 2013) adopted SysML, and used the following diagrams: Sequence, Package, Use Case, State Machine, and Definition Block. As the Sequence, Package, Use Case, and State Machine diagrams are part of the UML, and they were also included in Figure 6.

Finally, diagrams mentioned from SoaML were Service Architecture, Participant, and Data Exchange.

3.3.4 Q4 - In Which domains were ISO/IEC/IEEE 42010:2011 used?

As depicted in Figure 7, the Transportation domain is the subject of study in six articles. Transportation is an essential domain because it moves goods and people; for instance, in the United States 5 million tons of goods are transported each day. Thus, good software architecture is necessary for managing software-intensive systems in this domain (SELECTUSA, 2019). The second domain in which the standard is applied is Industry. The industry of vehicles produced over 70 million passenger cars in 2018 and software plays a big role in modern vehicles with dozens of sensors and actuators (OICA - International Organization of Motor Vehicle Manufacturers, 2018).

Figure 7 – Q4 - Domains in which the Architecture Description was applied.



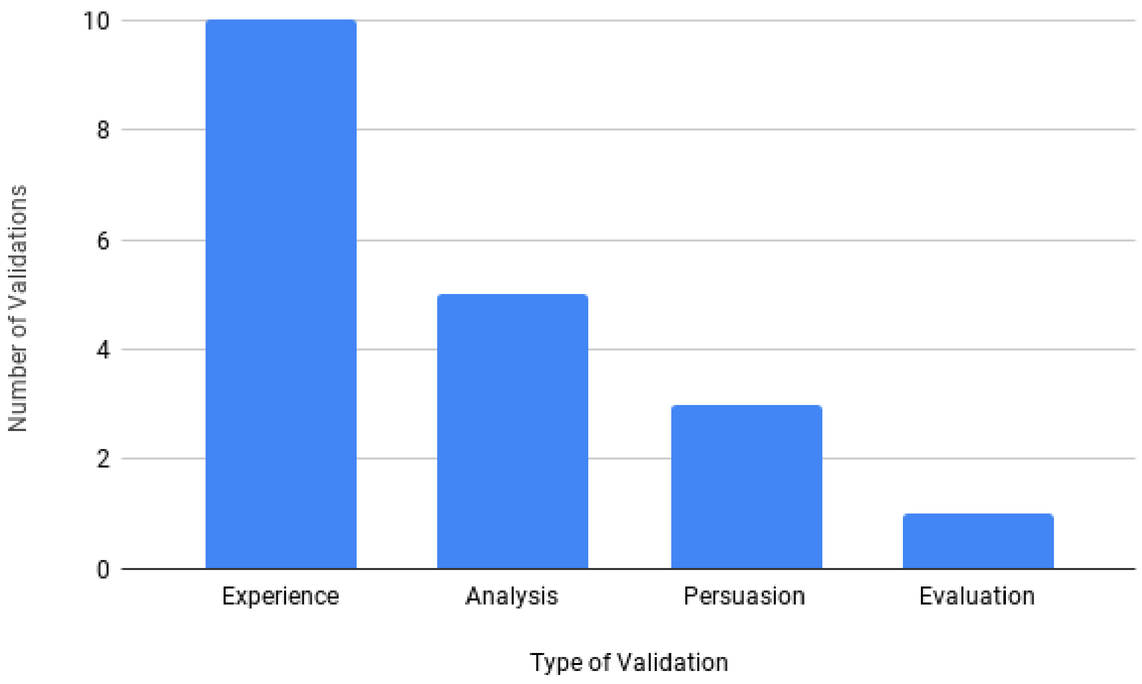
ISO/IEC/IEEE 42010:2011 was also applied in the Energy and Health domain. For the rest of the domains mentioned in Figure 7, the standard was mentioned once. Furthermore, we could not identify the domain of paper (DAS, 2016).

3.3.5 Q5 - What type of validation was considered?

In Software Engineering, the type of validation plays a big role in a good software project. The research presented in paper (SHAW, 2003) reports that the most successful types of validation were based on Analysis and real-world Experience. Figure 8 shows that Experience is the most used type of validation. Thus, one way to know if a software project will be successful is by testing it, preferably in the real world. Besides, if the project fails, it is also useful to know the problems that caused it.

The second most used type of validation is Analysis with five papers. Analysis is also a good type of validation because the authors can show if their solution is worth or not by applying the solution in realistic examples.

Figure 8 – Q5 - Types of Validations found in the papers.



3.3.6 Q6 - What type of research strategy was considered?

According to Johannesson e Perjons (2014), a research strategy is an overall plan for conducting a research study, and the strategy guides a researcher in planning, executing, and monitoring the study. Among the types of research strategy proposed in paper (JOHANNESSON; PERJONS, 2014), three types were found in 15 papers of this master thesis. Besides, it was not possible to define the research strategy of 4 papers.

Case Study was found in the majority of papers, in a total of 8 studies. This may be the research strategy near to the real world, which requires less financial resources. However, the fact that the research is successful in one case does not mean that it will be in any case. Action Research is the second considered one with five papers. This research strategy is used to solve problems in a real environment, and most of them were found in industry, health, financial, and transportation domains. The last one is simulation, found in the Energy domain.

Based on the definitions offered in (JOHANNESSON; PERJONS, 2014), the authors could not classify the research in papers (DAS, 2016) and (GÓMEZ et al., 2017), since they did not provide enough information about the type of research strategy adopted in these papers. Additionally, papers (WILLIAMS; STRACENER, 2013) and (VIDONI; VECCHIETTI, 2016) did not explicitly provide a research strategy, but in their future work, they will consider a case study for their research.

3.4 Threats to Validity

The main focus of Mapping studies are primary studies, and the results of this research may have been affected by threats to validity regarding Selection Bias, Data extraction, Generalization, and External validity to take into account during the research process.

3.4.1 Selection Bias

Some studies may be included or excluded into the SMS incorrectly. With the purpose of reducing this threat, the research protocol was discussed between the researchers to guarantee a common understanding, and the researchers made decisions together in a meeting for each item that they were in doubt.

3.4.2 Data extraction

Bias or problem of extraction can influence classifications and the analysis of selected studies. For reducing this bias, the researchers discussed the definitions of each item used to answer the questions. The data extraction form was a spreadsheet composed of 16 columns (data extraction attributes), which was structured as follows: five columns related to article identification, two columns related to the context of the research, and nine columns related to attributes of ISO/IEC/IEEE 42010:2011.

3.4.3 Generalization

The information from this research is possible to generalize. Even though the number of papers addressing ISO/IEC/IEEE 42010:2011 standard can be considered small, the search string was wide as possible, to reach the majority of papers in the databases.

4

ArchCaMo: A Maturity Model for Software Architecture

In order to propose the Architecture Capability Model (ArchCaMo), it was brought together a set of best practices to support the systematic and gradual evolution for the description of Software Architectures. The model proposed in this dissertation extends the best practices initially gathered in ISO/IEC/IEEE 42010: 2011, the results of a survey that investigated the use of the standard, and a systematic mapping study, in which the use of the standard in academic works is evaluated.

Therefore, in order to provide a short, medium and long term view of the proper functioning of how to describe a Software Architecture, a Maturity Model is created, consisting of five levels of maturity to be reached by organizations. Each level of maturity is described by the **Name**, **Description**, and the **Reference in the Standard**.

4.1 Procedure to propose the ArchCaMo Levels

Instead of establishing in rigid way activities, processes, and tasks to be followed, in ArchCaMo, the idea is to structure the levels in such a way that the organization can decide its processes for each given practice to be executed at each level.

The ArchCaMo's levels are based on, mostly, ISO/IEC/IEEE 42010:2011, a Systematic Mapping Study (SMS) executed by the authors described in Chapter 3, and also the authors' own experience in software architecture.

First, almost all the levels of the Maturity Model are linked to a topic of the standard. Therefore, the authors did not change the standard, but they have organized it in such a way that the architecture team can document the software/system's architecture in a sequential manner. Therefore, the software architecture team needs to set up its own method for executing the activities and consult the standard in case of any doubt during the documentation process in the software life-cycle.

The second reference for ArchCaMo's levels is the SMS. As a result, according to this classification and the knowledge/experience about the standard, it was proposed that the most used features of ISO/IEC/IEEE 42010:2011 would be considered in the initial levels and so on.

4.2 ArchCaMo Levels

The following subsections describe in detail the maturity levels and their respective requirements. Each level creates a basis for processes to the next level. An architectural description can only be considered within a level if it:

- Provides all requirements of the level in question;
- Provides all requirements of all dimensions and all previous levels;

The general characteristics for all levels of maturity are presented in the following sections.

4.2.1 Level 1 - Initial

The first level, Level 1, classified as initial from the architectural point of view, refers to all companies/organizations that do not have a formalized way to define the software architecture.

In this context, actions may be implemented to assure documentation quality, such as revisions to verify that the document conforms to the implemented architecture and monitoring it for nonconformities. However, these actions are carried out on an occasional (or *ad hoc*) basis, without systematic coordination between team members and without defining specific roles for quality assurance. In some projects, these actions are implemented because of the customer's request or at the individual initiative of one of the team members, or even the architect, concerned with quality of documentation.

Being at the initial level does not mean that the architecture does not exist, or it is not properly implemented as specified in the project. It indicates that it has not been documented in accordance with the ISO/IEC/IEEE (2011) standard or the involved stakeholders cannot understand it according to their business view.

Some basic process are not defined, so the software architecture team may document some aspects that they need or think that is necessary for the project.

Usually, there is a lack of training for professionals in the architecture team for the use of techniques and especially in tools. Many organizations recognize that there is a need to develop a more organized, and most professional, architectural documentation process (HEESCH; AVGERIOU; HILLIARD, 2012).

4.2.2 Level 2 - Minimum Architecture

The second level is made up of the most relevant aspects that are necessary to describe software architecture. This information, showed in Table 5, is of fundamental importance for an organization that intends to achieve Level 2, so it has clearly defined objectives and policies aligned in the architecture documentation with the general policies of the organization.

The following tables describe each level consisting of KAI (Key Architecture Item), the description of the KAI, and the respective reference in ISO/IEC/IEEE 42010:2011.

Table 5 – Level 2 - Minimum Architecture

KAI	**Description**	**Reference in the Standard**
2.1	Environment is identified	Item 4.2
2.2	Systems of Interest are identified	Item 5.2
2.3	Supplementary information is identified	Item 5.2
2.4	Stakeholders are identified	Item 5.3
2.5	Concerns for each Stakeholder are identified	Item 5.3
2.6	Viewpoints are defined	Item 5.4
2.7	Multiple Views are defined	Item 5.5
2.8	Models are defined	Item 5.6

Table 5 describes the Minimum Architecture, which is the name of Level 2. This table is based on the Context of architecture description and Conceptual model of an architecture description of ISO/IEC/IEEE 42010:2011.

4.2.3 Level 3 - Defined Architecture

After achieving Level 2 of maturity, the organization becomes able to advance to Level 3 of maturity. At this point, there is a number of maturity characteristics that have been deployed in Level 2.

For Level 3, the organization needs to use the information collected at the previous level and relate them, as well as manage each decision taken and describe why any one of them was taken. When all this information is collected, the organization has the fundamental characteristics to describe a software architecture completely. Table 6 describes KAIs related to the level Defined Architecture.

4.2.4 Level 4 - Improved Architecture

The two upper levels, Levels 4 and 5, are related to aspects that are needed to improve a software architecture situation in the life cycle that is already structured but still has possibilities of improvement.

Table 6 – Level 3 - Defined Architecture

KAI	**Description**	**Reference in the Standard**
3.1	Correspondence rules are identified	Item 5.7.2
3.2	Rationales for decisions are identified	Item 5.8.1
3.3	Concerns related to each decision are defined	Item 5.8.2
3.4	Decisions are managed	Item 5.8.2

In order for the organization to achieve Level 4, it must have a software quality policy with the organizational expectations for the process products and process itself. The organization policy should also establish periodic audits and how to handle the inconsistencies found during these audits.

In addition, the software architecture team needs to be knowledgeable and proficient in the use of architecture frameworks as well as architectural description tools in order to speed the description process in a healthy way.

Table 7 describes KAIs that compose the Consistent Architecture, based on the Conceptual model of an architecture framework, and the Conceptual model of an architecture description language of ISO/IEC/IEEE 42010:2011.

Table 7 – Level 4 - Consistent Architecture

KAI	**Description**	**Reference in the Standard**
4.1	Known inconsistencies are recorded	Item 5.7.1
4.2	Use of Architecture Frameworks	Item 6.1
4.3	Use of Architecture description languages	Item 6.3

4.2.5 Level 5 - Quantified and Evaluated Architecture

At this level of maturity, the processes are already sufficiently institutionalized, and there are guarantees of a right level of quality of the generated products.

The first point is the definition of metrics, so providing a quantitative control of the architecture description. Examples of control metrics are Timeline metrics (e.g., milestones, resources, pending tasks) and Artifact metrics (e.g., completeness, adherence to policies). Therefore, by analyzing these metrics, corrective actions will be taken if necessary. In addition, by formalizing the objectives of quality control, it is possible to measure and therefore control them, making clear the points that should receive more attention.

Being the highest level of maturity, it is already possible to move the focus from the actions of improving the results of the projects to the process of architectural description itself and find ways to optimize it. Moreover, it becomes important to measure the description process itself to find points where it can be improved, consequently making all the software architecture description projects even more efficient.

Finally, when this level of maturity is reached, the architecture leader can define an architecture group by analyzing all activities made by each component and verifying the specialty of each one.

Table 8 describes KAIs that compose the Consistent Architecture, based on the Conceptual model of an architecture framework, and the Conceptual model of an architecture description language of ISO/IEC/IEEE 42010:2011.

Table 8 – Level 5 - Quantified and Improved Architecture

KAI	Description	Reference in the Standard
5.1	Metrics on elements of the other levels are defined	Item 7
5.2	An Architecture Group is established	Page 2[1]

4.3 ArchCaMo Processes for each level

For each level, at least one architectural process is defined. These processes are performed mostly by the architect or team of software architects that are responsible for the software product.

First of all, it is supposed that not all organizations have a defined process for delimiting, structuring, and evaluating their software architecture. Even though some organizations that have a defined process may not fulfill all the KAI's, it does not mean that their architectural processes and products are irrelevant or insufficient.

4.3.1 Level 2 - Minimum Architecture

The Minimum Architecture, as explained before, is composed of eight key basic points that have to be identified and documented. At this particular level, any qualified person of the architecture team can identify these KAI's, but, in the end, the architect has to revise and approve.

- **P2.1** - Environment is identified. The environment is the context of determining the setting and circumstances of all influences upon a system. There are many environments, not only the place where the team is coding, but also, for instance, the operational environment, development environment, test environment, and so on. Additionally, when defining the environment, this will generate non-functional requirements and establish rules for the system.

[1] Architecting takes place in the context of an organization ("person or a group of people and facilities with an arrangement of responsibilities, authorities, and relationships.")(ISO/IEC/IEEE, 2011)

- **P2.2** - System-of-Interest is identified. The system in which life cycle is under consideration, and whose architecture is under consideration in the preparation of an architecture description. One way for identifying the system is the environment in which the system is.
- **P2.3** - Supplementary information is identified. This material shall be specified by the organization or project, which may include: date of issue and status; authors, reviewers, approving authority, or issuing organization; change history; summary; scope; context; glossary; version control information; configuration management information and references.
- **P2.4** - Stakeholders are identified. Each stakeholder is defined and ranked for their relative importance for that specific software product. In consonance with ISO/IEC/IEEE (2011), these stakeholders shall be considered, and when applicable, identified in the architecture description: users of the system, operators of the system, acquirers of the system, owners of the system, suppliers of the system, developers of the system, builders of the system and maintainers of the system.
- **P2.5** - Concerns for each Stakeholder are identified. For each stakeholder, its relative concerns are identified. In the ISO/IEC/IEEE (2011) documentation, one can find concerns that should be identified when applicable.
- **P2.6** - Viewpoints are defined. Through the interpretation of architectures views, viewpoints may be provided. The viewpoints will be used to frame specific concerns.
- **P2.7** - Multiple Views are defined. Each one of the multiple Views for describing the software architecture is defined to address the concerns held by one or more of its stakeholders.
- **P2.8** - Models are defined. From each viewpoint, models are defined using modeling conventions established by the architecture team, appropriate to the concerns to be addressed. Models are used to answer questions about the system-of-interest.

4.3.2 Level 3 - Defined Architecture

When the organization starts the identification of Level 3 KAI's, the goal of this level is making relations with the information collected from the previous level. Moreover, handling decisions that will be the basis of the architecture in question is necessary.

- **P3.1** - Correspondence rules are identified. These correspondences are used for indicating relations between two or more architecture models, and the rules are used to establish constraints on two or more architecture models.
- **P3.2** - Rationales for decisions are identified. Rationales are written in a list, and the reasons regarding each decision are documented for future reference.
- **P3.3** - Concerns related to each decision are defined.

- **P3.4** - Decisions are managed. Each decision is identified and written using a template defined by the architecture team. All decisions are identified, discussed, and written. At least a spreadsheet is used, but it is better to use a specific software system to keep track of each entry.

4.3.3 Level 4 - Improved Architecture

In Level 4, the focus is to improve the architecture by, first, finding discrepancies about what it should be and what it is. A tighter connection between development and execution to ensure errors are detected and fixed as soon as possible, and guarantee software quality is fundamental (FITZGERALD; STOL, 2017). Therefore, these found errors will be reviewed and recorded by the team, and it will help future software projects. In addition, the use of frameworks (optional) expedites the project, resulting in time gain for the entire project.

Finally, using an architecture description language is required, so a standard is established, and there is a common language for the whole team.

- **P4.1** - Known inconsistencies are recorded. An architecture description should record the inconsistencies and include an analysis of the consistency of its architecture models and its views.
- **P4.2** - Use of Architecture Frameworks. If the project is using an Architecture Framework, it should include conditions of applicability.
- **P4.3** - Use of Architecture Description Languages. The architecture description language should support all the aspects of the system-of-interest.

4.3.4 Level 5 - Quantified and Evaluated Architecture

At the highest level, the basic metrics of the architecture documentation process must be defined and also how they will be collected, analyzed, who and when they will be collected. These metrics also include architectural design metrics (examples: cost, effort). Finally, after all this effort, the architect will have enough information with the aim of organizing an Architecture group.

- **P5.1** - Metrics on elements of the other levels are defined. The software architect evaluates costs and benefits from the metrics, comparing to the other architectures that used ArchCaMo. For each architectural decisions, the architect analyzes the implications, extracting a number of metrics from it.
- **P5.2** - An Architecture Group is established, which is responsible for getting information of state of the art on software architecture. The group seeks to improve the architecture by

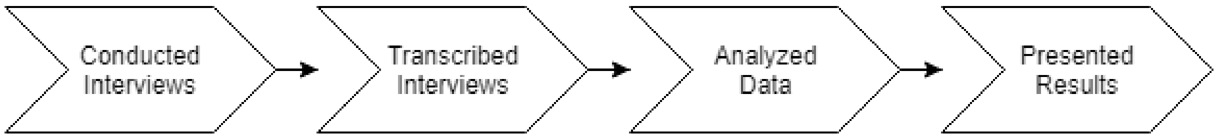
receiving input from many sources, including research conferences on software architecture, books, articles, and reports. The Architecture Group is also responsible for searching for improvements, as for instance, new methods, techniques, languages, processes, and so on, on the theme of software architecture and try to bring these approaches to be deployed in the organization.

5 ArchCaMo Evaluation Process

This chapter aims to make an assessment of the architecture documentation process in some organizations. A case study was conducted with organizations, being refined in research questions and specifying the metrics, in order to answer the listed questions quantitatively. An online questionnaire was prepared, with hard copies in case of internet inaccuracy, as well as the provision of the standard ISO/IEC/IEEE 42010: 2011, and a confidentiality term. It was suggested that the evaluation should be conducted, preferably, by someone from the software architecture area in the company, being open to the participation of other collaborators. The evaluation was conducted in person, through a visit to the companies, so data were collected by the researcher himself, with control of the answers and in case of doubts of the interviewee. Initial contact via e-mail with some companies was made, but only three were willing to answer the questionnaire.

The questionnaire was designed to be self-explanatory and was sized to be answered within 40 to 60 minutes. The defined questions aimed to meet the objective of the survey, addressing the following aspects: 1) respondent data, 2) standard items. Hence, the researches analyzed the results verifying the most used items of ISO/IEC/IEEE 42010:2011, as depicted in Figure 9.

Figure 9 – Research Methodology adopted for case study



5.1 Results from the questionnaire based on ISO/IEC/IEEE 42010:2011

The checklist has been taken directly from Sections 5, 6 and 7 of ISO/IEC/IEEE 42010 (ISO/IEC/IEEE, 2011), and it consists of statements. The responses are pre-defined and based on the Likert scale (JAMIESON, 2004), being 1- Totally Disagree, 2- Disagree, 3- Neither agree nor disagree, 4- Agree, and 5- Totally Agree.

Table 9 – Preliminary Checklist for ISO/IEC/IEEE 42010 Standard - Part 01

	Section 5 - Architecture descriptions	**A**	**B**	**C**
1	The architecture description has identified the system-of-interest and included supplementary information as determined by the project and/or organization.	4	4	3
2	The detailed content of identifying and supplementary information items has been specified by the organization and/or project.	4	2	3
3	There are any results from any evaluations of the architecture or its architecture description. (If yes, it shall be included).	3	4	2
4	The system stakeholders having concerns considered fundamental to the architecture of the system-of-interest have been identified.	4	4	4
5	The concerns considered fundamental to the architecture of the system-of-interest have been identified.	4	3	3
6	Each identified concern associates with the identified stakeholders having that concern have been identified.	4	4	3
7	Each architecture viewpoint has been included in the used architecture description.	2	3	2
8	Each included architecture viewpoint has been specified in accordance with the provisions of Table 11.	4	4	2
9	Each concern has been identified in accordance with items 4, 5 and 6 of table 9 framed by at least one viewpoint.	4	4	2
10	The architecture description has included exactly one architecture view for each architecture viewpoint used.	4	3	2
11	Each architecture view has adhered to the conventions of its governing architecture viewpoint.	4	2	2
12	There are of each architecture view the identification and supplementary information as specified by the organization and/or project.	4	4	2
13	There are of each architecture view the identification of its governing viewpoint.	2	4	2
14	There are of each architecture view architecture models that address all of the concerns framed by its governing viewpoint and cover the whole system from that viewpoint.	4	2	2
15	There are of each architecture view recording of any known issues within a view with respect to its governing viewpoint.	4	3	2
16	The architecture description may include information not part of any architecture view (Optional).	2	2	2
17	The architecture view is composed of one or more architecture models.	3	4	2
18	Each architecture model includes a identification version as specified by the organization and/or project.	4	4	2

Continued on next page

Table 9 – continued from previous page

	Section 5 - Architecture descriptions	**A**	**B**	**C**
19	Each architecture model identifies its governing model kind and adhered to the conventions of that model kind?.	3	4	2
20	There is any architecture model as part of more than one architecture view.	4	4	2
21	The architecture description has recorded any known inconsistencies across its architecture models and its views.	2	4	2
22	The architecture description has included an analysis of consistency of its architecture models and its views.	4	3	2
23	There are correspondences and correspondence rules, as specified in the items 24, 25, 26, 27 and 28, used to express, record, enforce and analyze consistency between models, views and other AD elements within an architecture description. (Optional).	4	4	2
24	Each correspondence in an architecture description has been identified and identify its participating AD elements.	4	3	2
25	If viewpoints and model kinds were defined, additional kinds of AD elements were introduced.	3	2	2
26	Each correspondence in an architecture description identifies any correspondence rules governing it.	4	4	2
27	The architecture description includes each correspondence rule applying to it.	3	3	2
28	For each identified correspondence rule, the architecture description has recorded whether the rule holds or otherwise recorded all known violations.	2	3	2
29	The architecture description includes a rationale for each architecture viewpoint included in terms of its stakeholders, concerns, model kinds, notations and methods.	5	3	2
30	The architecture description includes rationale for each decision considered to be a key architecture decision.	4	3	3
31	The architecture description provides evidence of the consideration of alternatives and the rationale for the choices made.	4	4	2
32	The architecture description records architecture decisions considered to be key to the architecture of the system-of-interest.	4	4	3

5.1.1 Organization A

A mixed economy organization (public and private), responsible for studies, projects, and execution of water supply, sewage, and sanitation works. The company has a service called Virtual Agency, which offers convenience when the customer needs access to some of the company's services, such as duplicate bills, reporting water leaks, requesting a water connection, among others.

The IT sector has about 20 employees, so it can be considered small for a company of this size. This sector has adopted the approach that any software that is not strategic to the company is outsourced, for instance, the commercial sector adopted GSAN (Integrated System of Management of Sanitation Services), a system from the Federal Government. GSAN is a system, developed with open source tools, of Commercial Operations Management and Internal Service Execution Control tools, freely available to Brazilian sanitation service providers and to their users. This system was created with the goal of raising the level of performance and efficiency of water supply and sewage collection companies, and it can be adapted to small, medium, and large companies.

The company was founded in the 1960's; therefore it has some legacy systems. When interviewing the software architect, he indicated that most legacy systems have already been updated, and the main goal of the company, in the near future, is to update all systems for adapting to new demands.

Considering the answers received from the survey, ArchCaMo is applied in this organization. As a result, Organization A was classified on level 1. For this organization to be at level 2, some procedures will be required. This organization already identifies the Environment (KAI 2.1), System of Interest (KAI 2.2), Supplementary Information (KAI 2.3), Stakeholders (KAI 2.4), and Concerns for each Stakeholder (KAI 2.5). When examining the answers of the respondent, the author perceived that there are conflicts among some questions, for example, questions 7 and 8. Therefore, assuming that the organization already conforms to the first 5 KAI's, the first missing information, according to the answers given by the interviewed, are the Viewpoints (KAI 2.6). There are non-documented views (KAI 2.7), which is the work-product itself, so the organization needs to write out what they have developed. Thereafter, viewpoints can be explicitly described. According to Harrison (2018), there is a related viewpoint that describes every view in a system, at least implicitly. Therefore, the company needs, basically, to document what already exists and, if necessary, look into other perspectives to add more viewpoints. Subsequently, this company will identify and document the Models (KAI 2.8) as, according to (ISO/IEC/IEEE, 2011), an architecture view, which is already identified, is composed of one or more architecture models.

5.1.2 Organization B

This organization is a Brazilian private higher education institution. The IT sector has about 50 employees, including System Analysts, Programmers, DataBase Analysts, and Software Testers.

According to the interviewed, the IT sector of the company has been doing some changes to focus on strategic activities. Therefore, any software that is not strategic to the company, such as the library system, warehouse system, payment system, is outsourced, so the employees can focus on what is vital for the company. Another example is the wireless network maintenance and installation, so the University hired a specialized company for this function because there is a constant expansion in the networks, and it was overloading the local employees. On the other hand, the academic software, which is the target of the University, is developed in house. For example, there is a system focused on student activity management as well as a portal for students to track their performance during their academic life. Currently, there is a team focused on the mobile version of the system, so the system is being improved constantly due to the high demand of students for this system.

Table 10 – Preliminary Checklist for ISO/IEC/IEEE 42010 Standard - Part 02

	Section 6 - Architecture frameworks and ADL	**A**	**B**	**C**
1	The architecture framework (AF) has included information identifying the architecture framework.	4	4	2
2	The AF has included the identification of one or more concerns (Items 4, 5 e 6 from table 9).	4	4	2
3	The AF has included the identification of one or more stakeholders having those concerns (Items 4,5 e 6 from table 9).	5	4	2
4	The AF has included one or more architecture viewpoints that frame those concerns (table 11).	4	4	2
5	The AF has included any correspondence rules (Items 21 and 28 from table 9).	1	2	2
6	The AF has included conditions of applicability.	4	4	2
7	The AF has establish its consistency with the provisions of the conceptual model in 4.2.	3	4	2
8	Each applicable stakeholder identified in the AF has been considered and identified in the architecture description (Items 4, 5 e 6 of table 9).	4	4	3
9	Each applicable concern identified in the AF has been considered and identified in the architecture description (Items 4, 5 e 6 of table 9).	4	4	2
10	Each applicable viewpoint specified by the AF (Items 1, 2 e 3 of table 10) is included (Items 7, 8 e 9 from table 9) in the architecture description.	3	3	2
11	Each applicable correspondence rule specified by the AF is included in the architecture description (Items 27 e 28 of table 9).	3	3	2
12	The architecture description conforms to the requirements of table 9.	4	3	2
13	The AF may establish additional rules for adherence.	4	4	2
14	The ADL has specified the identification of one or more concerns to be expressed by the ADL (Items 4, 5 e 6 from table 9).	4	3	2
15	The ADL has specified the identification of one or more stakeholders having those concerns (Items 4, 5 e 6 of table 9).	5	4	2
16	The ADL has specified the model kinds implemented by the ADL which frame those concerns (according to item 4 of table 11).	4	4	2
17	The ADL has specified any architecture viewpoints (according to table 11)?	3	4	2
18	The ADL has specified correspondence rules (Items 21 and 28 of table 9) relating its model kinds on item 16 of this table.	4	3	2

After evaluating Organization B, through the given answers, this organization was classified on level 1. According to the answers, this company already identifies the Environment (KAI 2.1), System of Interest (KAI 2.2) and Stakeholders (KAI 2.4). The first point that is not documented is the Supplementary Information (KAI 2.3). This material can be found by consulting stakeholders (authors, reviewers), and by verifying the version controls. The second element is the Concerns for each Stakeholder (KAI 2.5). In consonance with the answers, it did not identify all the fundamental concerns, but at least it was associated with the identified concerns. According to the interviewed, for some projects, there is a short deadline, so defining all concerns is not possible because of time. However, views and viewpoints are expressed through concerns, so a lack of fundamental concerns will generate architectural smells on the project. Besides, two points of ArchCaMo, architecture view and viewpoints, will be compromised. Thus, the organization just needs to consult all the fundamental stakeholders and get the missing information. The last three elements, Viewpoints (KAI 2.6), Multiple Views (KAI 2.7), and Models (KAI 2.8), can be identified by following the same processes described in Section 5.1.1.

Table 11 – Preliminary Checklist for ISO/IEC/IEEE 42010 Standard - Part 03

	Section 7 - Architecture Viewpoints	**A**	**B**	**C**
1	The architecture viewpoint has specified one or more concerns framed by this viewpoint (Items 4, 5 e 6 from table 1).	4	4	2
2	The architecture viewpoint has specified typical stakeholders for concerns framed by this viewpoint.	5	3	2
3	The architecture viewpoint has specified one or more model kinds used in this viewpoint.	2	4	2
4	The architecture viewpoint has specified for each model kind identified in the question above, the languages, notations, conventions, modelling techniques, analytical methods and/or other operations to be used on models of this kind.	2	4	2
5	The architecture viewpoint has specified references to its sources.	4	4	2
6	The architecture viewpoint has included information on architecting techniques used to create, interpret or analyze a view governed by this viewpoint?	4	4	2

5.1.3 Organization C

This organization is a public organization in the field of justice. The IT sector has 34 employees. Currently, they started to adopt software solutions from other institutions of the same field in the country.

In the IT sector, there are two types of software development: the first one is for simple solutions, as queue management for people service; and the second one is a software that has been developed in this organization. This software has been adopted in the whole country and needs to be maintained frequently. For this software, there is a team composed of people of other institutions.

Organization C does not have one chief software architect. Instead, a team of 3 senior software analysts is responsible for the software architecture. As depicted in Table 9, this institution basically does not implement most of the software architect concepts. The only concept that the organization implements is the identification of stakeholders. Additionally, there are some items that are performed partially. Actually, the organization is improving and updating its software documentation in general, because they are already experiencing the impacts of TD.

After Applying ArchCaMo to evaluate Organization C, the company is classified on level 1. When analyzing the answers, this company already determines the Environment (KAI 2.1), System of Interest (KAI 2.2), and Stakeholders (KAI 2.4). Therefore, for this organization to be at level 2, some procedures will be required. First, determining the Supplementary Information (KAI 2.3) by consulting Stakeholders, documents, and control versions. Thereafter, the concerns for each stakeholder have to be identified and associated (KAI 2.5). Some stakeholders' interests may help the project in terms of time, but others may weak it or even stop it. Besides, even among stakeholders from the same group, there may be conflicting concerns. Finally, the last three elements, Viewpoints (KAI 2.6), Multiple Views (KAI 2.7), and Models (KAI 2.8), can be established by attending the same steps described in Section 5.1.1.

Table 12 shows the points that are fully fulfilled (●), the points that are partially fulfilled

(◒), and that are still absent (○) for each organization to reach level 2.

Table 12 – Level 2 - Minimum Architecture - Achieved Points

KAI	**Description**	**A**	**B**	**C**
2.1	Environment is identified	●	●	●
2.2	Systems of Interest are identified	●	●	●
2.3	Supplementary information is identified	●	○	◒
2.4	Stakeholders are identified	●	●	●
2.5	Concerns for each Stakeholder are identified	●	◒	◒
2.6	Viewpoints are defined	○	◒	○
2.7	Multiple Views are defined	○	◒	○
2.8	Models are defined	○	○	○

5.2 Threats to Validity

As well as Systematic Mapping Studies, Surveys are also subject to threats to validity.

5.2.1 Internal

Survey respondents may have a negative influence on the results according to their experience. Besides, respondents may not have understood some questions, and to mitigate this risk, the questionnaire was applied under the supervision of one of the researchers, who could help to understand the questions, making the research faster and less susceptible to errors.

5.2.2 External

There is no possibility of generalization of the obtained information since the empirical analysis was performed in an environment that does not represent a significant population of software development organizations, with only three respondents.

5.2.3 Construct

The range of answers was five, based on the Likert Scale, so there is a tendency of the respondents not to mark extremes, as can be seen in all answers.

5.2.4 Bias of Conclusions

The conclusions presented by the researchers may have been influenced by the different degrees of researchers' experience. This risk was mitigated by continuous discussions among researchers, and also by considering experienced software developers, all with more than ten years of experience.

6 Conclusion

This chapter presents the analysis of the objectives, limitations of the work, the main contributions of this master thesis, suggestions for future works, and the author's conclusions.

This master thesis begins with the presentation of its general context, the research problem, and its relevance to the area. In addition, the objectives and methodology applied to the research are also presented. Then, a literature review is described, where the main themes that involve the work are presented: Software Architecture, technical debt, the standard ISO/IEC/IEEE 42010:2011, and Maturity Models.

Moreover, state of the art on the use of ISO/IEC/ IEEE 42010:2011 through a Systematic Mapping Study (SMS) is presented in Chapter 3. The SMS investigates studies which used ISO/IEC/IEEE 42010:2011 on their software architectures, and which parts of it was most used. Also, this study was conducted by following guidelines provided in (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). The authors identified 139 papers, after the exclusion criteria and full reading, 119 papers were rejected, so in total, 19 studies were identified as relevant and analyzed. Each paper was evaluated with a quality assessment. Then, the findings were quantitatively classified according to a publication year, publication channel, the used aspects of ISO/IEC/IEEE 42010:2011, domains, and type of validation. The included studies were published between 2012 and 2018. The year 2018 received the most significant amount of publications, six papers. Also, fourteen papers were published in conferences. The main aspects considered from ISO/IEC/IEEE 42010:2011 are Stakeholders and Concerns. The most mentioned model types were the UML Use Case, Class, and State Machine diagrams, and the most cited domain was transportation. The type of validation Experience is the most used one, found in 10 papers, and the most common research strategy was the case study.

After presenting all the theoretical consolidation for the research, the Architecture Capability Model (ArchCaMO) is proposed: a maturity model for documentation and evaluation of software architectures. The ArchCaMo was defined based on the standard ISO/IEC/IEEE

42010:2011. The main objective of this model is to minimize the problems of designing software architecture and organize the way of describing it. The five presented levels of ArchCaMO are based on the described rules of ISO/IEC/IEEE 42010:2011. The initial level, Level 1, considered as unstable, refers to all companies/organizations that do not have a formalized way to define the software architecture. Levels 2 and 3 are made up of the most relevant aspects that are necessary to describe a software architecture. Levels 4 and 5 are related to aspects that are needed to improve a software architecture situation in the life cycle that is already structured. After description of the levels, how the levels work through the process is described.

Finally, a case study was applied to 3 organizations, one public, other private, and the third one, a mixed economy. All the interviewed organizations were classified on level 1. ArchCaMo can help these three companies to improve their software architecture processes. The first step is the classification of them based on the survey. Then, when the responsible architect knows where the software architecture description is classified, he/she needs to start gathering missing information. In these particular cases, a large number of software systems are already in production, but there is a lack of software architecture documentation. The maturity model does not define a method for collecting information. Thus the architecture team can choose the best way that fits in their software processes. However, some data has to be collected in a specific order because part of ArchCaMo processes may be a result of this previous information, or even it is the result of liking it. For example, viewpoints are identified through views, and correspondence rules may be the result of connecting rationales to stakeholder's concerns. ArchCaMO maturity model evaluates software architectures, providing a direction to the software architecture team of how to manage a description of software architectures from new and legacy systems.

6.1 Contributions

The results obtained in this master's dissertation present some practical contributions (industry-oriented) and some contributions to the research itself (academic oriented). Thus, the following contributions resulting from this research are highlighted bellow:

1. Industry oriented:
   - A maturity model that orients how documenting a software architecture (Chapter 4);
   - Studies with professionals working in the industry aiming to identify improvements to the proposed model (Chapter 5);
2. Academic oriented:
   - State of the art about the use of the standard ISO/IEC/IEEE 42010:2011 (Chapter 3);
   - A detailed description of the process for proposing the maturity model (Chapter 4);

These contributions were shared with the academic and industry community through publications written throughout the development of this research.

6.2 Limitations

Although this master thesis clearly defines that its main contribution is focused on the development of a maturity model for Software Architectures description, focusing on the evaluation and improvement of their documentation, there are possible limitations related to this master thesis.

6.2.1 Systematic Mapping Study

One of the limitations of this Systematic Mapping Study is the possibility of bias. The advisor of this work has extensive experience in software engineering and architecture, helping on the selection of the studies. Disagreements over the inclusion/exclusion of a particular study and data extraction were resolved by referring to the original study and discussing it to establish consensus between the reviewers. Relevant studies may not be included, mainly studies produced in educational institutions, not published in events or periodicals, although attempts have been made to minimize this possibility utilizing automatic searches in paper indexing mechanisms.

6.2.2 Maturity Model

The maturity model proposed in this master thesis also presents, as one of its main limitations, the absence of complementary components that support its application in a real environment. Among these components, the current needs to be defined:

- A set of metrics and indicators to monitor, evaluate, suggest, decide, or change the course of a process or set of activities aimed to achieve a goal, in this case, the advance levels.
- A glossary that addresses the key terms related to Software Architecture, Technical Debt and Maturity models, for specific words and unfamiliar expressions to the Software Architecture Team;
- A best practice database;
- A database on which the histories of the evaluations performed would be recorded;
- Conduct Studies with more companies in other countries and in different domains.

6.3 Future Researches

This research work generated as its main result the conception of ArchCaMo. This model should be used to support the systematic and gradual evolution of software architecture documentation maturity in organizations. Although the general objective and specific objectives have been achieved, there are still some suggestions and recommendations for future work for this research:

1. Replication of the case study on architectural documentation, in the context of other companies using other frameworks, in order to verify if data lead to similar results to the one presented here by means of the 3 case studies.
2. Develop an implementation guide to facilitate the use/adoption of the proposed model. This guide will help when applying the model as the basis for a specific assessment, or certification in software architecture documentation;
3. Create an online self-assessment tool to provide indications of the organization's level of maturity/capability and to suggest possible improvements. This tool would aim to assist the documentation quality assurance activities, making it possible to bring together in a single tool the management of nonconformities, lessons learned, reports, among other features that collaborate with the work of those responsible for quality assurance, in a mature environment;
4. A post-mortem analysis to evaluate the architectural documentation maturity of completed projects and correlate maturity levels with the quality of the developed system.
5. An assessment by Software Architecture experts, both from academia and industry, on the model. This assessment aims to obtain feedback about advantages and disadvantages with the use of the proposed model.
6. Document the decisions of each item at each level, justifying them with more information, either through industry experience or referencing works from other authors.
7. Expert assessment of the evaluation process used in this master thesis. This item has the goal to give more credibility to the process used to evaluate the proposed maturity model.

6.4 Thesis-Related Publications

This section presents the contributions related to this master's thesis.

6.4.1 Published Papers

1. JÚNIOR, Ademir AC; MISRA, Sanjay; SOARES, Michel S. A Systematic Mapping Study on Software Architectures Description Based on ISO/IEC/IEEE 42010: 2011. In: International Conference on Computational Science and Its Applications. Springer, Cham, 2019. p. 17-30. Qualis A3.

2. JÚNIOR, Ademir AC; MISRA, Sanjay; SOARES, Michel S. ArchCaMO-A Maturity Model for Software Architecture Description Based on ISO/IEC/IEEE 42010: 2011. In: International Conference on Computational Science and Its Applications. Springer, Cham, 2019. p. 31-42. Qualis A3.

6.4.2 Submitted Papers

1. Checklist based on ISO/IEC/IEEE 42010 for Finding Technical Debts in a Defined Software Architecture. Journal Computer Standards & Interfaces. Qualis A4.

Bibliography

Abrahamsson, P.; Babar, M. A.; Kruchten, P. Agility and architecture: Can they coexist? *IEEE Software*, v. 27, n. 2, p. 16–22, March 2010. Cited 2 times on pages 16 and 17.

AHMED, F.; CAPRETZ, L. F. An Architecture Process Maturity Model of Software Product Line Engineering. *Innovations in Systems and Software Engineering*, v. 7, n. 3, p. 191–207, 2011. Cited on page 26.

ALLMAN, E. Managing Technical Debt. *Queue*, ACM, New York, NY, USA, v. 10, n. 3, p. 10:10–10:17, mar. 2012. Cited on page 28.

ALVES, N. S. R. et al. Towards an Ontology of Terms on Technical Debt. In: *2014 Sixth International Workshop on Managing Technical Debt*. [S.l.: s.n.], 2014. p. 1–7. Cited 2 times on pages 28 and 29.

AMIN, M. S.; BLACKBURN, T.; GARSTENAUER, A. Deploying a Recall Mitigation Framework for Systems Engineering. *Engineering Management Journal*, 30, n. 1, p. 42–56, 2018. Cited on page 35.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. 3rd. ed. [S.l.]: Addison-Wesley Professional, 2012. Cited 2 times on pages 15 and 21.

BERTOLINO, A. et al. Use Case Description of Requirements for Product Lines. In: *International Workshop on Requirements Engineering for Product Lines*. [S.l.: s.n.], 2002. v. 2002, p. 12. Cited on page 37.

BOOCH, G. The Economics of Architecture-First. *IEEE Software*, v. 24, n. 5, p. 18–20, 2007. Cited 2 times on pages 15 and 16.

BOSCH, J. *Design and use of software architectures: adopting and evolving a product-line approach*. [S.l.]: Pearson Education, 2000. Cited on page 16.

BOSCH, J. Software Architecture: The Next Step. In: *European Workshop on Software Architecture*. [S.l.]: Springer, 2004. (EWSA 2004), p. 194–199. Cited on page 21.

BROWN, N. et al. Managing Technical Debt in Software-reliant Systems. In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. New York, NY, USA: ACM, 2010. (FoSER '10), p. 47–52. Cited on page 28.

BUCHGEHER, G.; WEINREICH, R.; KRIECHBAUM, T. Making the Case for Centralized Software Architecture Management. In: *SWQD*. [S.l.]: Springer, 2016. (Lecture Notes in Business Information Processing, v. 238), p. 109–121. Cited on page 16.

CAPILLA, R. et al. 10 Years of Software Architecture Knowledge Management: Practice and future. *Journal of Systems and Software*, v. 116, p. 191–205, 2016. Cited on page 15.

CHAABANE, M.; BOUASSIDA, I.; JMAIEL, M. System of Systems Software Architecture Description Using the ISO/IEC/IEEE 42010 Standard. In: *Proceedings of the Symposium on Applied Computing*. New York, NY, USA: ACM, 2017. (SAC '17), p. 1793–1798. Cited on page 35.

CHAUDRON, M. R. V. Empirical Studies into UML in Practice: Pitfalls and Prospects. In: *2017 IEEE/ACM 9th International Workshop on Modelling in Software Engineering (MiSE)*. [S.l.: s.n.], 2017. p. 3–4. Cited on page 36.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. *CMMI for Development: Guidelines for Process Integration and Product Improvement*. 3rd. ed. [S.l.]: Addison-Wesley Professional, 2011. Cited on page 26.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. *CMMI for development: Guidelines for Process Integration and Product Improvement*. [S.l.]: Pearson Education, 2011. Cited on page 27.

CRICHTON, R. et al. An Architecture and Reference Implementation of an Open Health Information Mediator: Enabling Interoperability in the Rwandan Health Information Exchange. In: SPRINGER. *International Symposium on Foundations of Health Informatics Engineering and Systems*. [S.l.], 2012. p. 87–104. Cited 3 times on pages 24, 35, and 36.

CUNNINGHAM, W. The WyCash Portfolio Management System. *SIGPLAN OOPS Mess.*, ACM, New York, NY, USA, v. 4, n. 2, p. 29–30, dez. 1992. Cited on page 28.

DAS, A. Context-aware Architecture Utilizing Computing with Words and ISO/IEC/IEEE 42010. In: *SoutheastCon 2016*. [S.l.: s.n.], 2016. p. 1–6. Cited 3 times on pages 35, 38, and 39.

DÍAZ-PACE, J. A. et al. Producing Just Enough Documentation: An Optimization Approach Applied to the Software Architecture Domain. *Journal on Data Semantics*, v. 5, n. 1, p. 37–53, Mar 2016. Cited on page 17.

DIJKSTRA, E. W. The Structure of THE-multiprogramming System. *Commun. ACM*, v. 26, n. 1, p. 49–52, 1968. Cited on page 15.

DING, L.; MEDVIDOVIC, N. Focus: A light-weight, incremental approach to software architecture recovery and evolution. In: *Proceedings Working IEEE/IFIP Conference on Software Architecture*. [S.l.: s.n.], 2001. p. 191–200. Cited 2 times on pages 21 and 22.

DING, W. et al. How Do Open Source Communities Document Software Architecture: An Exploratory Survey. In: *2014 19th International Conference on Engineering of Complex Computer Systems*. [S.l.: s.n.], 2014. p. 136–145. Cited on page 17.

EFFENBERGER, F.; HILBERT, A. Towards an Energy Information System Architecture Description for Industrial Manufacturers: Decomposition & Allocation View. *Energy*, v. 112, p. 599 – 605, 2016. Cited 3 times on pages 24, 35, and 36.

ERNST, N. A. et al. Measure It? Manage It? Ignore It? Software Practitioners and Technical Debt. In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. New York, NY, USA: ACM, 2015. (ESEC/FSE 2015), p. 50–60. Cited on page 27.

FALESSI, D. et al. Applying Empirical Software Engineering to Software Architecture: Challenges and Lessons Learned. *Empirical Software Engineering*, Springer, v. 15, n. 3, p. 250–276, 2010. Cited 4 times on pages 15, 16, 26, and 30.

FALESSI, D. et al. Decision-making Techniques for Software Architecture Design: A Comparative Survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 43, n. 4, p. 1–33, out. 2011. Cited on page 21.

FITZGERALD, B.; STOL, K.-J. Continuous software engineering: A Roadmap and Agenda. *Journal of Systems and Software*, v. 123, p. 176 – 189, 2017. Cited on page 47.

FONTANA, F. A.; FERME, V.; ZANONI, M. Towards Assessing Software Architecture Quality by Exploiting Code Smell Relations. In: *2015 IEEE/ACM 2nd International Workshop on Software Architecture and Metrics*. [S.l.: s.n.], 2015. p. 1–7. Cited on page 29.

FRANÇA, J. M.; LIMA, J. d. S.; SOARES, M. S. Development of an Electronic Health Record Application using a Multiple View Service Oriented Architecture. *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems*, v. 2, p. 308–315, 2017. Cited 3 times on pages 24, 35, and 36.

GALSTER, M.; TAMBURRI, D. A.; KAZMAN, R. Towards Understanding the Social and Organizational Dimensions of Software Architecting. *SIGSOFT Softw. Eng. Notes*, v. 42, n. 3, p. 24–25, 2017. Cited on page 16.

GARCIA, J. et al. Identifying Architectural Bad Smells. In: *Proceedings of the 2009 European Conference on Software Maintenance and Reengineering*. Washington, DC, USA: IEEE Computer Society, 2009. p. 255–258. ISBN 978-0-7695-3589-0. Cited on page 29.

GARCIA, J. et al. Toward a Catalogue of Architectural Bad Smells. In: MIRANDOLA, R.; GORTON, I.; HOFMEISTER, C. (Ed.). *Architectures for Adaptive Software Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 146–162. Cited on page 29.

GARLAN, D. Software Architecture: A Roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*. [S.l.: s.n.], 2000. (ICSE ’00), p. 91–101. Cited on page 15.

GARLAN, D. Software Architecture: A Travelogue. In: *Proceedings of the on Future of Software Engineering*. [S.l.: s.n.], 2014. (FOSE 2014), p. 29–39. Cited 2 times on pages 15 and 17.

GIL, A. C. *Como Elaborar Projetos de Pesquisa*. 4. ed. [S.l.]: Atlas, 2002. ISBN 8522431698. Cited on page 18.

GÓMEZ Álvaro et al. Development and Implementation of a High-Level Control System for the Underwater Remotely Operated Vehicle VISOR3. *IFAC-PapersOnLine*, v. 50, n. 1, p. 1151 – 1156, 2017. 20th IFAC World Congress. Cited 3 times on pages 35, 36, and 39.

GRAAF, K. et al. How Organisation of Architecture Documentation Affects Architectural Knowledge Retrieval. *Science of Computer Programming*, v. 121, p. 75–99, 2016. Cited on page 17.

GUESSI, M. et al. A Systematic Literature Review on the Description of Software Architectures for Systems of Systems. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2015. (SAC ’15), p. 1433–1440. Cited on page 27.

GUO, Y.; SEAMAN, C. A Portfolio Approach to Technical Debt Management. In: *Proceedings of the 2Nd Workshop on Managing Technical Debt*. New York, NY, USA: ACM, 2011. (MTD ’11), p. 31–34. Cited on page 30.

HARRISON, R. *TOGAF(r) 9 Foundation Study Guide*. [S.l.]: Van Haren, 2018. Cited on page 52.

HEESCH, U. V.; AVGERIOU, P.; HILLIARD, R. A Documentation Framework for Architecture Decisions. *Journal of Systems and Software*, v. 85, n. 4, p. 795 – 820, 2012. Cited 3 times on pages 24, 35, and 42.

HILLIARD, R. et al. On the Composition and Reuse of Viewpoints across Architecture Frameworks. In: *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*. [S.l.: s.n.], 2012. p. 131–140. Cited on page 35.

HOFMEISTER, C.; NORD, R.; SONI, D. *Applied Software Architecture*. [S.l.]: Addison-Wesley Professional, 2000. Cited on page 16.

HOLVITIE, J. et al. Technical Debt and Agile Software Development Practices and Processes: An Industry Practitioner Survey. *Information and Software Technology*, v. 96, p. 141 – 160, 2018. Cited on page 28.

HUTCHINSON, J.; WHITTLE, J.; ROUNCEFIELD, M. Model-Driven Engineering Practices in Industry: Social, Organizational and Managerial Factors that Lead to Success or Failure. *Science of Computer Programming*, v. 89, p. 144 – 161, 2014. Cited on page 36.

"IEEE Recommended Practice for Architectural Description for Software-Intensive Systems". *IEEE Std 1471-2000*, p. 1–30, Oct 2000. Cited on page 15.

"ISO/IEC 12207: 2008(E) IEEE Std 12207-2008 - Redline: Systems and Software Engineering – Software Life Cycle Processes - Redline". [S.l.]: IEEE, 2008. Cited on page 16.

"ISO/IEC 15288: 2008(E) IEEE Std 15288-2008 (Revision of IEEE Std 15288-2004) - Redline: ISO/IEC/IEEE International Standard - Systems and software engineering System life cycle processes - Redline". [S.l.]: IEEE, 2008. Cited on page 16.

ISO/IEC/IEEE. Systems and Software Engineering – Architecture Description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, p. 1–46, 1 2011. Cited 9 times on pages 22, 24, 25, 27, 42, 45, 46, 50, and 52.

JAMIESON, S. Likert scales: How to (Ab)Use Them. *Medical Education*, v. 38, n. 12, p. 1217–1218, 2004. Cited on page 50.

JAZAYERI, M.; RAN, A.; LINDEN, F. van der. *Software Architecture for Product Families: Principles and Practice*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. Cited on page 22.

JOHANNESSON, P.; PERJONS, E. *An Introduction to Design Science*. [S.l.]: Springer, 2014. Cited 3 times on pages 19, 32, and 39.

JOHN, I.; MUTHIG, D. Tailoring Use Cases for Product Line Modeling. In: *Proceedings of the International Workshop on Requirements Engineering for product lines*. [S.l.: s.n.], 2002. v. 2002, p. 26–32. Cited on page 37.

KANNENGIESSER, U.; MüLLER, H. Towards Viewpoint-Oriented Engineering for Industry 4.0: A Standards-Based Approach. In: *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. [S.l.: s.n.], 2018. p. 51–56. Cited on page 35.

KARKHANIS, P.; BRAND, M. G.; RAJKARNIKAR, S. Defining the C-ITS Reference Architecture. In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. [S.l.: s.n.], 2018. p. 148–151. Cited on page 24.

KARKHANIS, P.; BRAND, M. G. J. V. D.; RAJKARNIKAR, S. Defining the C-ITS Reference Architecture. In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. [S.l.: s.n.], 2018. p. 148–151. Cited 3 times on pages 35, 36, and 37.

KAVAKLI, E. et al. WiP: An Architecture for Disruption Management in Smart Manufacturing. In: *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. [S.l.: s.n.], 2018. p. 279–281. Cited 2 times on pages 35 and 36.

KLIMKO, G. Knowledge Management and Maturity Models: Building Common understanding. In: BLED, SLOVENIA. *Proceedings of the 2nd European Conference on Knowledge Management*. [S.l.], 2001. p. 269–278. Cited on page 27.

KOHLEGGER, M.; MAIER, R.; THALMANN, S. *Understanding Maturity Models. Results of a Structured Content Analysis*. [S.l.]: na, 2009. Cited on page 27.

KOUROSHFAR, E. et al. A Study on the Role of Software Architecture in the Evolution and Quality of Software. *Proceedings of the 12th Working Conference on Mining Software Repositories*, p. 246–257, 2015. Cited on page 21.

KREGER, H. et al. *IBM Advantage for Service Maturity Model Standards*. 2009. <https://www.ibm.com/developerworks/webservices/library/ws-OSIMM/index.html>. Accessed: 2019-09-24. Cited on page 27.

KRUCHTEN, P.; NORD, R. L.; OZKAYA, I. Technical Debt: From Metaphor to Theory and Practice. *IEEE Software*, v. 29, n. 6, p. 18–21, Nov 2012. Cited 2 times on pages 28 and 29.

KRUCHTEN, P. et al. Technical Debt: Towards a Crisper Definition Report on the 4th International Workshop on Managing Technical Debt. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 38, n. 5, p. 51–54, ago. 2013. Cited 2 times on pages 27 and 28.

KRUCHTEN, P.; OBBINK, H.; STAFFORD, J. The Past, Present, and Future for Software Architecture. *IEEE Software*, v. 23, n. 2, p. 22–30, 2006. Cited 2 times on pages 15 and 21.

KRUCHTEN, P. B. The 4+1 View Model of Architecture. *IEEE Software*, IEEE, v. 12, n. 6, p. 42–50, Nov 1995. Cited on page 16.

LAKATOS, E. M.; MARCONI, M. d. A. *Fundamentos de Metodologia Científica*. 8. ed. [S.l.]: Atlas, 2017. Cited on page 18.

LAPES, L. o. R. o. S. E. *StArt - State of the Art through Systematic Review*. 2018. <http://lapes.dc.ufscar.br/tools/start_tool>. Online; accessed 8-June-2018. Cited on page 33.

LEHMAN, M. M.; BELADY, L. A. *Program Evolution: Processes of Software Change*. [S.l.]: Academic Press Professional, Inc., 1985. Cited on page 29.

LI, Z.; AVGERIOU, P.; LIANG, P. A Systematic Mapping Study on Technical Debt and its Management. *Journal of Systems and Software*, v. 101, p. 193 – 220, Dec 2015. Cited 3 times on pages 27, 28, and 29.

LIM, E.; TAKSANDE, N.; SEAMAN, C. A Balancing Act: What Software Practitioners Have to Say about Technical Debt. *IEEE Software*, v. 29, n. 6, p. 22–27, Nov 2012. Cited on page 28.

LIPPERT, M.; ROOCK, S. *Refactoring in Large Software Projects: Performing Complex Restructurings Successfully*. Chichester: John Wiley & Sons, 2006. Cited on page 29.

MACIA, I. et al. Are Automatically-detected Code Anomalies Relevant to Architectural Modularity?: An Exploratory Analysis of Evolving Systems. In: *Proceedings of the 11th Annual International Conference on Aspect-oriented Software Development*. New York, NY, USA: ACM, 2012. (AOSD '12), p. 167–178. Cited on page 29.

MAIER, A. M.; MOULTRIE, J.; CLARKSON, P. J. Assessing Organizational Capabilities: Reviewing and Guiding the Development of Maturity Grids. *IEEE Transactions on Engineering Management*, v. 59, n. 1, p. 138–159, Feb 2012. Cited on page 27.

MARANZANO, J. F. et al. Architecture Reviews: Practice and Experience. *IEEE Software*, v. 22, n. 2, p. 34–43, March 2005. Cited on page 22.

MARTINI, A.; BOSCH, J.; CHAUDRON, M. Investigating Architectural Technical Debt Accumulation and Refactoring Over Time: A Multiple-Case Study. *Information and Software Technology*, v. 67, p. 237 – 253, 2015. Cited on page 27.

MAY, G. et al. An Approach to Development of System Architecture in Large Collaborative Projects. In: SPRINGER. *IFIP International Conference on Advances in Production Management Systems*. [S.l.], 2017. p. 67–75. Cited 2 times on pages 35 and 36.

MCCONNELL, S. *Managing Technical Debt*. 2008. <http://www.construx.com/uploadedfiles/resources/whitepapers/Managing%20Technical%20Debt.pdf>. Accessed: 2019-08-07. Cited on page 28.

MEYER, M.; HELFERT, M.; O'BRIEN, C. An Analysis of Enterprise Architecture Maturity Frameworks. In: GRABIS, J.; KIRIKOVA, M. (Ed.). *Perspectives in Business Informatics Research*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 167–177. Cited on page 26.

MO, J.; BECKETT, R. Architectural Knowledge Integration in a Social Innovation Context. *Advances in Transdisciplinary Engineering*, v. 7, p. 763–772, 2018. Cited on page 35.

MOHAGHEGHI, P.; APARICIO, M. E. An Industry Experience Report on Managing Product Quality Requirements in a Large Organization. *Information and Software Technology*, Elsevier, v. 88, p. 96–109, 2017. Cited on page 16.

MUSIL, J. et al. An Architecture Framework for Collective Intelligence Systems. In: *2015 12th Working IEEE/IFIP Conference on Software Architecture*. [S.l.: s.n.], 2015. p. 21–30. Cited 2 times on pages 24 and 35.

NORD, R. L. et al. In Search of a Metric for Managing Architectural Technical Debt. In: *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*. [S.l.: s.n.], 2012. p. 91–100. Cited on page 29.

OBERGFELL, P. et al. Viewpoint-Based Methodology for Adaption of Automotive E/E-Architectures. In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. [S.l.: s.n.], 2018. p. 128–135. Cited on page 35.

OICA - International Organization of Motor Vehicle Manufacturers. *2018 PRODUCTION STATISTICS*. 2018. <http://www.oica.net/category/production-statistics/2018-statistics/>. Online; accessed 9-July-2019. Cited on page 38.

ONLINE, C. D. *Maturity*. 2019. <https://dictionary.cambridge.org/pt/dicionario/ingles/maturity>. Online; accessed 4-July-2019. Cited on page 26.

OQUENDO, F. Software Architecture Challenges and Emerging Research in Software-Intensive Systems-of-Systems. In: *Software Architecture - 10th European Conference, ECSA 2016, Copenhagen, Denmark, November 28 - December 2, 2016, Proceedings*. [S.l.: s.n.], 2016. p. 3–21. Cited on page 15.

OSTADZADEH, S. S.; SHAMS, F. Towards a Software Architecture Maturity Model for Improving Ultra-Large-Scale Systems Interoperability. *CoRR*, abs/1401.5752, 2014. Cited on page 26.

PANUNZIO, M.; VARDANEGA, T. An Architectural Approach with Separation of Concerns to Address Extra-Functional Requirements in the Development of Embedded Real-Time Software Systems. *Journal of Systems Architecture*, v. 60, n. 9, p. 770 – 781, 2014. Cited 2 times on pages 35 and 36.

PARNAS, D. L. A Technique for Software Module Specification with Examples. *Communications of the ACM*, v. 15, n. 5, p. 330–336, maio 1972. Cited on page 15.

PARNAS, D. L. On the Criteria to Be Used in Decomposing Systems into Modules. *Communications of the ACM*, v. 15, n. 12, p. 1053–1058, 1972. Cited on page 15.

PATTON, M. Q. Qualitative research. In: ____. *Encyclopedia of Statistics in Behavioral Science*. [S.l.]: American Cancer Society, 2005. Cited on page 18.

PAULK, M. C. et al. *Capability Maturity Model for Software (Version 1.1)*. Pittsburgh, PA, 1993. Cited on page 26.

PERRY, D. E.; WOLF, A. L. Foundations for the Study of Software Architecture. *ACM SIGSOFT Software engineering notes*, ACM, v. 17, n. 4, p. 40–52, 1992. Cited on page 21.

PETERSEN, K. et al. Systematic Mapping Studies in Software Engineering. In: *EASE*. [S.l.: s.n.], 2008. v. 8, p. 68–77. Cited on page 18.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. *Information and Software Technology*, v. 64, p. 1–18, 2015. Cited 2 times on pages 31 and 56.

PRESSMAN, R. S.; MAXIM, B. *Software Engineering: A Practitioner's Approach*. [S.l.]: McGraw-Hill Education, 2014. Cited on page 21.

SANTOS, R. d. S. *Metodologia Científica: A Construção do Conhecimento*. [S.l.]: Lamparina, 2007. Cited on page 19.

SCHUMACHER, A.; EROL, S.; SIHN, W. A Maturity Model for Assessing Industry 4.0 Readiness and Maturity of Manufacturing Enterprises. *Procedia CIRP*, v. 52, p. 161 – 166, 2016. The Sixth International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016). Cited on page 26.

SEAMAN, C.; GUO, Y. Chapter 2 - Measuring and Monitoring Technical Debt. In: ZELKOWITZ, M. V. (Ed.). [S.l.]: Elsevier, 2011, (Advances in Computers, v. 82). p. 25 – 46. Cited 2 times on pages 29 and 30.

SELECTUSA. *LOGISTICS AND TRANSPORTATION SPOTLIGHT*. 2019. <https://www.selectusa.gov/logistics-and-transportation-industry-united-states>. Online; accessed 9-July-2019. Cited on page 37.

SHAW, M. Writing Good Software Engineering Research Papers. In: *25th International Conference on Software Engineering, 2003. Proceedings.* [S.l.: s.n.], 2003. p. 726–736. Cited 2 times on pages 32 and 38.

SHULL, F. Perfectionists in a World of Finite Resources. *IEEE Software*, v. 28, n. 2, p. 4–6, March 2011. Cited 2 times on pages 28 and 30.

TOFAN, D. et al. Past and Future of Software Architectural Decisions – A Systematic Mapping Study. *Information and Software Technology*, v. 56, n. 8, p. 850–872, 2014. Cited on page 15.

VIDONI, M.; VECCHIETTI, A. Towards a Reference Architecture for Advanced Planning Systems. In: HAMMOUDI, S. et al. (Ed.). *Proceedings of the 18th International Conference on Enterprise Information Systems, Vol. 1 (ICEIS)*. [S.l.: s.n.], 2016. p. 433–440. Cited 4 times on pages 24, 35, 36, and 39.

WATERMAN, M.; NOBLE, J.; ALLAN, G. How Much Up-front?: A Grounded Theory of Agile Architecture. In: *Proceedings of the 37th International Conference on Software Engineering - Volume 1*. [S.l.: s.n.], 2015. (ICSE '15), p. 347–357. Cited on page 17.

WILLIAMS, J. L.; STRACENER, J. T. First steps in the development of a Program Organizational Architectural Framework (POAF). *Systems Engineering*, v. 16, n. 1, p. 45–70, 2013. Cited 4 times on pages 24, 35, 37, and 39.

WIRFS-BROCK, R.; YODER, J.; GUERRA, E. Patterns to Develop and Evolve Architecture During an Agile Software Project. In: *Proceedings of the 22Nd Conference on Pattern Languages of Programs*. [S.l.: s.n.], 2015. (PLoP '15), p. 9:1–9:18. Cited on page 17.

YANG, C.; LIANG, P.; AVGERIOU, P. A Systematic Mapping Study on the Combination of Software Architecture and Agile Development. *Journal of Systems and Software*, v. 111, p. 157–184, 2016. Cited on page 16.

ZACHMAN, J. A. A Framework for Information Systems Architecture. *IBM Systems Journal*, IEE, v. 26, p. 276–292, 1987. Cited on page 16.

Appendix

APPENDIX A – Formulário ISO/IEC/IEEE 42010:2011

Formulário ISO/IEC/IEEE 42010:2011

Você está sendo convidado(a) como voluntário(a) a participar de uma pesquisa sobre Arquitetura de Software. Esta pesquisa tem como objetivo a extração de dados conforme parâmetros da norma ISO/IEC/IEEE 42010.

Participação do estudo – A minha participação no referido estudo será de responder um questionário, que levará o tempo médio de 40 minutos.

Sigilo e Privacidade – Estou ciente de que a minha privacidade será respeitada, ou seja, meu nome ou qualquer dado ou elemento que possa, de qualquer forma, me identificar será mantido em sigilo. Os pesquisadores se responsabilizam pela guarda e confidencialidade dos dados.

Autonomia – É assegurada a assistência durante toda a pesquisa, bem como me garantido o livre acesso a todas as informações e esclarecimentos adicionais sobre o estudo e suas consequências, enfim, tudo que eu queira saber antes, durante e depois da minha participação. Declaro que fui informado de que posso me recusar a participar do estudo, ou retirar meu consentimento a qualquer momento, sem precisar justificar, e de, por desejar, sair da pesquisa, não sofrerei qualquer prejuízo à assistência que venho recebendo.

Declaração – Declaro que li e entendi todas as informações presentes neste Termo. Enfim, tendo sido orientado quanto ao teor de todo o aqui mencionado e compreendido a natureza e o objetivo do já referido estudo, eu manifesto meu livre consentimento em participar, estando totalmente ciente de que não há nenhum valor econômico, a receber ou pagar, por minha participação.

Uso de Imagem - Não haverá divulgação de imagem, gravação ou áudio a terceiros.

Pesquisadores responsáveis:
Ademir Almeida de Costa Junior
Michel dos Santos Soares

1. **Endereço de e-mail** *

____

1 - Perfil Organização/Entrevistado

2. **1.1 Qual seu nível de Escolaridade?**
Marcar apenas uma oval.

- ◯ Superior Incompleto
- ◯ Superior Completo
- ◯ Especialização
- ◯ Mestrado
- ◯ Doutorado

3. **1.2 Tempo de Experiência em TI**

____

4. **1.3 Em que setor sua organização se enquadra?**
Marcar apenas uma oval.

- ◯ Pública
- ◯ Privada

5. **1.4 Atualmente qual cargo ocupado na organização?**

2 - ISO/IEC/IEEE 42010 - Descrições de Arquitetura

Para auxílio na compreensão da norma, o link abaixo contém a tradução da norma, bem como notas explicativas sobre as perguntas.

https://goo.gl/TeR5Fw

6. **2.1 A descrição da arquitetura identifica o sistema de interesse e inclui informações suplementares, conforme determinado pelo projeto e /ou organização.**
   Exemplos de informações suplementares em uma descrição de arquitetura são: data de emissão e status; autores, revisores, autoridade de aprovação, organização emissora; histórico; resumo; escopo; contexto; glossário; informações de controle de versão; informações de gerenciamento de configuração e referências.
   Marcar apenas uma oval.

   - Discordo totalmente
   - Discordo
   - Nem concordo nem discordo
   - Concordo
   - Concordo Totalmente

7. **2.2 O conteúdo detalhado de identificação e itens de informações suplementares estão conforme especificado pela organização e/ou projeto.**
   Marcar apenas uma oval.

   - Discordo totalmente
   - Discordo
   - Nem concordo nem discordo
   - Concordo
   - Concordo Totalmente

8. **2.3 Há algum resultado de alguma avaliação da arquitetura ou de sua descrição.**
   Marcar apenas uma oval.

   - Discordo totalmente
   - Discordo
   - Nem concordo nem discordo
   - Concordo
   - Concordo Totalmente

9. **2.4 Os stakeholders do sistema com expectativas consideradas fundamentais para a arquitetura do sistema de interesse são identificados.**
   Marcar apenas uma oval.

   - Discordo totalmente
   - Discordo
   - Nem concordo nem discordo
   - Concordo
   - Concordo Totalmente

10. **2.5 As expectativas consideradas fundamentais para a arquitetura do sistema de interesse foram identificadas.**

Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

11. **2.6 Cada expectativa identificada está associada aos stakeholders identificados com essa expectativa.**

Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

12. **2.7 Cada ponto de vista da arquitetura está incluído na descrição da arquitetura usada.**

Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

13. **2.8 Cada ponto de vista da arquitetura incluído está especificado de acordo com as disposições da Seção 4?**

Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

14. **2.9 Cada expectativa é identificada de acordo com 2.4, 2.5 e 2.6, enquadrada por pelo menos um ponto de vista?**

Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

15. **2.10 A descrição da arquitetura inclui exatamente uma visão de arquitetura para cada ponto de vista de arquitetura usado.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

16. **2.11 Cada visão de arquitetura está de acordo com as convenções de seu ponto de vista de arquitetura dominante.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

17. **2.12 Cada visão de arquitetura inclui a identificação e informação suplementar (2.1), conforme especificado pela organização e/ou projeto.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

18. **2.13 Cada visão de arquitetura inclui a identificação de seu ponto de vista dominante.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

19. **2.14 Cada visão de arquitetura inclui modelos de arquitetura que abordam todas as expectativas enquadradas por seu ponto de vista dominante e cobrem todo o sistema desse ponto de vista.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

20. **2.15 Cada visão de arquitetura inclui o registro de quaisquer problemas conhecidos em uma visão com relação ao ponto de vista dominante.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

21. **2.16 A descrição da arquitetura inclui informações que não fazem parte de nenhuma visão de arquitetura.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

22. **2.17 A arquitetura é composta de um ou mais modelos de arquitetura.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

23. **2.18 Cada modelo de arquitetura inclui a identificação da versão conforme especificado pela organização e/ou projeto.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

24. **2.19 Cada modelo de arquitetura identifica seu tipo de modelo de governança e adere às convenções desse tipo de modelo.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

25. **2.20 Algum modelo de arquitetura pode fazer parte de mais de uma visão de arquitetura.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

26. **2.21 A descrição da arquitetura registra quaisquer inconsistências conhecidas em seus modelos de arquitetura e suas visões.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

27. **2.22 A descrição da arquitetura incluiu uma análise da consistência de seus modelos de arquitetura e suas visões?**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

28. **2.23 Existem Correspondências e regras de correspondência, conforme especificado em 2.24 a 2.28, usadas para expressar, registrar, fazer cumprir e analisar a consistência entre modelos, visualizações e outros elementos da Descrição de Arquitetura dentro de uma descrição de arquitetura.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

29. **2.24 Cada correspondência e seus elementos participantes em uma descrição de arquitetura foram identificados.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

30. **2.25 Quando pontos de vista e tipos de modelo foram definidos, tipos adicionais de elementos de descrição de arquitetura foram introduzidos.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

31. **2.26 Cada correspondência em uma descrição de arquitetura identifica alguma regra de correspondência que a rege.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

32. **2.27 A descrição da arquitetura inclui cada regra de correspondência aplicada a ela.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

33. **2.28 Para cada regra de correspondência identificada, uma descrição de arquitetura registra se a regra é valida ou registra todas as violações conhecidas.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

34. **2.29 A descrição da arquitetura incluiu uma justificativa para cada ponto de vista de arquitetura incluído para uso (2.7 a 2.9) em termos de seus stakeholders, expectativas, tipos de modelo, notações e métodos.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

35. **2.30 A descrição da arquitetura incluiu justificativa para cada decisão considerada como uma decisão chave de arquitetura.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

36. **2.31 A descrição da arquitetura fornece evidências da consideração de alternativas e da justificativa para as escolhas feitas.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

37. **2.32 A descrição de arquitetura registra decisões de arquitetura consideradas essenciais para a arquitetura do sistema de interesse.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

3 - ISO/IEC/IEEE 42010 - Frameworks de arquitetura e linguagens de descrição de arquitetura

38. **3.1 O framework de arquitetura inclui informações que identificam o framework de arquitetura.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

39. **3.2 O framework de arquitetura inclui a identificação de uma ou mais expectativas (2.4 a 2.6).**
Marcar apenas uma oval.

- ⬭ Discordo totalmente
- ⬭ Discordo
- ⬭ Nem concordo nem discordo
- ⬭ Concordo
- ⬭ Concordo Totalmente

40. **3.3 O framework de arquitetura inclui a identificação de um ou mais stakeholders que tenham essas expectativas (2.4 a 2.6).**
Marcar apenas uma oval.

- ⬭ Discordo totalmente
- ⬭ Discordo
- ⬭ Nem concordo nem discordo
- ⬭ Concordo
- ⬭ Concordo Totalmente

41. **3.4 O framework de arquitetura inclui um ou mais pontos de vista de arquitetura que enquadram essas expectativas (4.1 a 4.6).**
Marcar apenas uma oval.

- ⬭ Discordo totalmente
- ⬭ Discordo
- ⬭ Nem concordo nem discordo
- ⬭ Concordo
- ⬭ Concordo Totalmente

42. **3.5 O framework de arquitetura inclui quaisquer regras de correspondência (por 2.12 a 2.18).**
Marcar apenas uma oval.

- ⬭ Discordo totalmente
- ⬭ Discordo
- ⬭ Nem concordo nem discordo
- ⬭ Concordo
- ⬭ Concordo Totalmente

43. **3.6 O framework de arquitetura inclui condições de aplicabilidade.**
Marcar apenas uma oval.

- ⬭ Discordo totalmente
- ⬭ Discordo
- ⬭ Nem concordo nem discordo
- ⬭ Concordo
- ⬭ Concordo Totalmente

44. **3.7 O framework da arquitetura estabeleceu a consistência com as disposições do modelo conceitual em 4.2.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

45. **3.8 Cada stakeholder aplicável identificado na estrutura de arquitetura foi considerado e identificado na descrição da arquitetura (2.4 a 2.6).**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

46. **3.9 Cada preocupação aplicável identificada no framework de arquitetura foi considerada e identificada na descrição da arquitetura (2.4 a 2.6).**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

47. **3.10 Cada ponto de vista aplicável é especificado pelo framework de arquitetura (3.1 a 3.7) incluída (2.7 a 2.9) na descrição da arquitetura.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

48. **3.11 Cada regra de correspondência aplicável é especificada pelo framework de arquitetura incluída na descrição da arquitetura (2.27 e 2.28).**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

49. **3.12 A descrição da arquitetura está em conformidade com os requisitos da seção 2.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

50. **3.13 O framework de arquitetura estabelece regras adicionais para aderência.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

51. **3.14 A linguagem de descrição da arquitetura especificou a identificação de uma ou mais expectativas a serem expressas pela Linguagem de Descrição de Arquitetura (2.4 a 2.6).**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

52. **3.15 A linguagem de descrição da arquitetura especificou a identificação de um ou mais stakeholders tendo essas expectativas (2.4 a 2.6).**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

53. **3.16 A linguagem de descrição da arquitetura especificou os tipos de modelo implementados pela linguagem de descrição da arquitetura que enquadram essas expectativas (de acordo com com a questão 4.4).**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

54. **3.17 A linguagem de descrição da arquitetura especificou algum ponto de vista de arquitetura (de acordo com a seção 4)?**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

55. **3.18 A linguagem de descrição da arquitetura especificou as regras de correspondência (por 2.12 a 2.18). relacionando seus tipos de modelo de acordo coma a 3.16?**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

4 - ISO/IEC/IEEE 42010 - Pontos de vista de arquitetura

56. **4.1 O ponto de vista da arquitetura especificou uma ou mais expectativas enquadradas por este ponto de vista (2.4 a 2.6).**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

57. **4.2 O ponto de vista da arquitetura especificou os stakeholders típicos para expectativas enquadradas por esse ponto de vista.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

58. **4.3 O ponto de vista da arquitetura especificou um ou mais tipos de modelo usados neste ponto de vista.**
Marcar apenas uma oval.

- Discordo totalmente
- Discordo
- Nem concordo nem discordo
- Concordo
- Concordo Totalmente

59. **4.4 O ponto de vista de arquitetura foi especificado para cada tipo de modelo identificado na questão acima, as linguagens, notações, convenções, técnicas de modelagem, métodos analíticos e/ou outras operações a serem usadas em modelos desse tipo.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

60. **4.5 O ponto de vista da arquitetura especificou referências a suas fontes.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente

61. **4.6 O ponto de vista da arquitetura incluiu informações sobre as técnicas de arquitetura usadas para criar, interpretar ou analisar uma visão governada por esse ponto de vista.**
Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Nem concordo nem discordo
- ◯ Concordo
- ◯ Concordo Totalmente